D I P L O M A R B E I T

Application Service Providing

mit Schwerpunkt auf Organisation und Technik

ausgeführt am Institut für Softwaretechnik
der Technischen Universität Wien

unter der Anleitung von
Prof. Dr. A Min Tjoa und **DI Robert Bruckner** als
verantwortlich mitwirkendem Universitätsassistenten

durch

Günther Walter

Auhofstr. 101-111 / 7 / 7
A-1130 Wien

Wien, 25. 3. 2002

Kurzfassung

Application Service Providing (ASP), das Mieten von Software per Internet, gewinnt immer mehr an Bedeutung. Die Anwendungen laufen hierbei vollständig am Server ab, nur die Benutzerschnittstelle wird zum Client übertragen.

Unternehmen sehen in dieser neuen Art des Anwendungsgebrauchs große Potentiale, ihre unternehmerische Effizienz durch Reduzierung von Komplexität, Kosten und Personal zu verbessern sowie die Markteintrittsphase zu verkürzen.

Kapitel 1 dieser Arbeit gibt einen umfassenden Überblick über das Thema ASP, beschreibt, wie ASP entstanden ist, welche Perspektiven und Risiken das ASP Modell bietet, welche Marktteilnehmer und Arten von Angeboten es gibt, sowie die wirtschaftlichen und rechtlichen Rahmenbedingungen.

Damit ist eine ideale Vorbereitung auf den Hauptteil dieser Arbeit in Kapitel 2 gegeben, wo organisatorische und technische Voraussetzungen und Konzepte für eine qualitative Anwendungsbereitstellung über das Internet vorgestellt und erläutert werden. Den Schwerpunkt bilden dabei die Anwendersicht und die ASP Basistechnologien.

Im praktischen Teil der Diplomarbeit in Kapitel 3 werden verschiedene ASP Basistechnologien aus der Praxis, nämlich traditionelle Internetsprachen, Microsoft .NET, BEA WebLogic, IBM WebSphere, Microsoft Windows Terminal Services und Citrix MetaFrame XPe beschrieben und auf Eignung für ASP untersucht.

In Kapitel 4 wird eine kleine ASP-Applikation namens "miiCurrencyConverterService" vorgestellt, die im Rahmen der Diplomarbeit bei der Firma "mii-marcus izmir informationsmanagement ag" entstanden ist.

Den Abschluss bildet im Anhang ein Interview über technische Aspekte von ASP mit Herrn Dominik PAIHA, Senior Consultant bei Microsoft Österreich.

Ziel dieser Arbeit ist es, die völlig veränderten Rahmenbedingungen von ASP im Unterschied zu Stand-Alone-Applikationen oder herkömmlichen Client-/Server-Applikationen darzustellen und aufzuzeigen, wie man den organisatorischen und technischen Herausforderungen von ASP in Theorie und Praxis begegnen kann.

Inhaltsverzeichnis

Kapitel 1 Einführung in ASP 5

1.1. Definition, Modell und Merkmale von ASP 5

1.2. Historie und Abgrenzung von ASP 7

1.2.1. IT – Outsourcing 8

1.2.2. Application hosting 9

1.2.3. Browser-based computing 10

1.3. Sichtweisen von ASP 10

1.3.1. Anwendersicht 11

1.3.2. ASP – Sicht 17

1.3.3. Technologiesicht 20

1.4. ASP – Player 21

1.4.1. Kernkompetenzen, Geschäftsmodelle und Produktkategorien 21

1.4.2. ASP - Player Schichtenmodell 22

1.4.3. ASP - Player Mengenmodell (Rollen) 24

1.4.4. Arten von Application Service Providern 25

1.5. Applikationen und Adressaten 27

1.6. Wirtschaftliche Aspekte 29

1.6.1. Wertschöpfungskette 29

1.6.2. Marktsituation und Marktentwicklung 30

1.6.3. Betriebswirtschaftliche Aspekte 32

1.7. Rechtliche Aspekte 34

1.7.1. Vertragseinordnung 35

1.7.2. Support- und Pflegeleistungen (SLAs) 35

1.7.3. Haftung und Gewährleistung 36

1.7.4. Lizenzrechtliche Aspekte 37

1.7.5. Datenschutz 37

Kapitel 2 Organisation und Technik 39

2.1. Software Basistechnologien für ASP 39

2.1.1. Terminalprogramme 39

2.1.2. Websprachen 41

2.1.3. Vergleich von Terminals und Websprachen 44

2.2. Internetfähigkeit und Kommunikation 45

2.2.1. Front-End Kommunikation 45

2.2.2. Middleware 46

2.2.3. Back-End Kommunikation 46

2.3. Mehrbenutzerfähigkeit und Zugriffsverwaltung 47

2.3.1. Mehrbenutzerfähigkeit 47

2.3.2. Zugriffsverwaltung 48

2.4. Plattformunabhängigkeit und Integration 49

2.4.1. Plattformunabhängigkeit ... 50
2.4.2. Integration ... 53

2.5. Verrechnungstechnologien ... 55
2.5.1. Verrechenbare Ereignisse in einer ASP-Umgebung ... 56
2.5.2. Komponenten eines ASP-Abrechnungssystems ... 57
2.5.3. Verrechnung von Applikationsnutzung ... 58

2.6. Server Herausforderungen ... 59
2.6.1. Server Architektur und Applikationsprogrammlogik ... 59
2.6.2. Verfügbarkeit ... 63
2.6.3. Performance und Skalierbarkeit ... 68
2.6.4. ASP Operation Management ... 72
2.6.5. Sicherheit ... 77

2.7. Client Herausforderungen ... 91
2.7.1. Benutzerinterface-Programmlogik ... 91
2.7.2. Benutzerfreundlichkeit und Anpassbarkeit ... 94
2.7.3. Mächtigkeit des Clients ... 102
2.7.4. Client Sicherheit ... 103

Kapitel 3 ASP Basistechnologien in der Praxis ... 107

3.1. Traditionelle Internettechnologien und -sprachen ... 107
3.1.1. Beschreibungssprachen ... 109
3.1.2. Clientseitige Internetsprachen ... 110
3.1.3. Serverseitige Internetsprachen ... 113
3.1.4. Eignung für ASP ... 118

3.2. Microsoft .NET ... 119
3.2.1. .NET Konzept ... 119
3.2.2. .NET Framework ... 120
3.2.3. Eignung für ASP ... 123

3.3. BEA WebLogic und IBM WebSphere ... 127
3.3.1. Java 2, Enterprise Edition ... 128
3.3.2. BEA WebLogic Server 6.1 ... 131
3.3.3. IBM WebSphere Server 4.0 ... 131
3.3.4. Java Application Server Verwaltungsfunktionen ... 132
3.3.5. Eignung für ASP und Vergleich mit Microsoft.NET ... 134

3.4. Microsoft Windows Terminal Services ... 136
3.4.1. WTS Konzept ... 136
3.4.2. Leistungsmerkmale von WTS und RDP 5.0 ... 137
3.4.3. WTS Server Verwaltungsfunktionen ... 138
3.4.4. Eignung für ASP ... 140

3.5. Citrix MetaFrame XP ... 142
3.5.1. Citrix MetaFrame XPe Konzept ... 143
3.5.2. Leistungsmerkmale von MetaFrame XPe und ICA ... 144
3.5.3. MetaFrame XPe Server Verwaltungsfunktionen ... 147
3.5.4. Eignung für ASP ... 149

3.6. Proprietäre Lösungen ... 151

3.7. Abschließende Ergebnisse ... 152

Kapitel 4 ASP am Beispiel von "miiCurrencyConverterService" ... 154

4.1. Funktionalität von MCCS ... 155

4.2. Einsatz von MCCS ... 157

4.2.1. Aufruf über HTTP-GET und HTTP-POST 157
4.2.2. Aufruf über SOAP 158
4.3. Programmierung von MCCS 159
4.4. Zusammenfassung MCCS 162
Kapitel 5 Zusammenfassung und Ausblick 164
Anhang Interview über technische Aspekte von ASP 168
Abkürzungsverzeichnis 178
Literaturverzeichnis 184

Kapitel 1

Einführung in ASP

Zweck dieser Einführung ist, auf folgende Fragen Antworten zu finden:

- Was ist ASP und wo liegen seine Ursprünge und Motivationen ?
- Welche Chancen und Risiken birgt ASP ?
- Wie funktioniert es ?
- Für wen bietet sich ASP an und warum ?

Trotz der Dynamik des ASP-Begriffs wird zunächst versucht, eine Definition von ASP zu geben und diese anhand der Darstellung seiner Merkmale verständlich zu machen. Es folgt eine Historie von ASP, die Ursprünge und Motivationen von ASP deutlich macht und gleichzeitig ASP von ähnlichen Verfahren abgrenzt.

Da am ASP-Geschäft mehrere Player wie Provider, Partner und Kunden beteiligt sind, die an diesem neuen Modell jeweils unterschiedliche Interessen an den Tag legen, wird anschließend eine Unterteilung in drei Sichtweisen (Anwendersicht, Application Service Provider-Sicht, Technologie-Sicht) vorgenommen.

Diese Struktur bietet sich an, da hierbei Modell, Funktionalität, Motivation, Chancen und Risiken von ASP sichtbar werden und eine ideale Vorbereitung auf das Schwerpunktkapitel "Organisation und Technik" gegeben ist.

1.1. Definition, Modell und Merkmale von ASP

Definition 1.1. :

***ASP** - **A**pplication **S**ervice **P**roviding:* Anwendungsfunktionalität *(Application Service)* wird, üblicherweise auf Mietbasis, von einem Rechenzentrum *(Application Service Provider)* über Internet oder ein privates Netzwerk mehreren Kunden zur Verfügung gestellt *(Providing).*

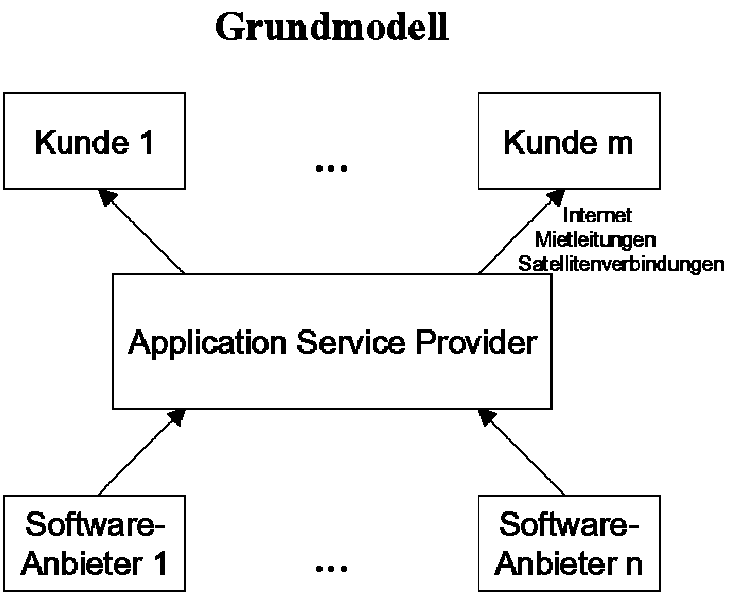

Abbildung 1-1 : Die Beziehungen eines ASPs zu Kunden und Softwareanbietern
[Stahlknecht 2000]

Bei ASP handelt es sich um ein neues und erfolgversprechendes Geschäftsfeld der IT-Branche: Anwender kaufen nicht mehr länger Software, sondern sie mieten sie von einem Application Service Provider (ASP), der ihnen die Anwendungen über das Internet bereitstellt. Die Anwendungen und Anwendungsdaten werden auf einem zentralen Server (Data Center) abgelegt und auch dort verwaltet. Der ASP ist verantwortlich für Betrieb, Bereitstellung, Management, Update und Pflege der Anwendungen und Daten. Die Anwendungsfunktionalität wird dabei dem Benutzer in Echtzeit über einen Internetbrowser oder über eine proprietäre Laufzeitumgebung zur Verfügung gestellt. Der Client übernimmt nur Usereingabeverarbeitungs-, Rendering- und Kommunikationsaufgaben, während der Server die eigentliche Anwendungsfunktionalität zur Verfügung stellt.

ASP und Anwender unterliegen einem vertraglichen Übereinkommen, genannt Service Level Agreement (SLA). Dabei verpflichtet sich der Provider, die vertraglich geregelten Leistungen zu erfüllen. Der Kunde bezahlt nach einem geeigneten Abrechnungsmodell (zeit-, benutzer- und/oder transaktionsbasiert).

Die eben genannten Merkmale des ASP-Modells geben die Grundlage für eine Charakterisierung des ASP Begriffs.

Nach dem Studium vieler Definitionen und Merkmalsbeschreibungen von ASP ([SCN Education 2000], [Cherry Tree 1999], [Diebold 1999], [Farleit 2000], [Tao 2000]) kristallisierten sich folgende sechs Eigenschaften heraus, die alle ASP-Lösungen gemeinsam haben:

- **Zentrale Speicherung, Verwaltung und Echtzeitbereitstellung von Applikationen und Daten**

  Applikationen laufen auf dem Server, werden vom ASP verwaltet und stellen ihre Funktionen dem Client "per request" zur Verfügung.

- **Webbasierter Zugriff**

  Der Zugriff erfolgt über das weltweite und standardisierte Internet oder über ein privates Netzwerk (z.B. über ein "virtual private network" (VPN)).

- **Konzentration aller Beteiligten auf Kernkompetenzen**

  Das ASP Modell ermöglicht Unternehmen, die Administration von IT-Aufgaben dem ASP zu überlassen. Der ASP selbst kooperiert mit Partnern und Lieferanten. Somit können sich alle beteiligten Player wie Provider, Partner des Providers und ASP-Kunden auf die Kernkompetenzen konzentrieren.

- **Minimale Anpassung an den Kunden**

  Ein und dieselbe Anwendung muss gleichzeitig dutzenden oder tausenden Benutzern zur Verfügung stehen. Daher ist die Anpassung der Software an den individuellen Kunden eher gering ausgeprägt, um sogenannte "economies of scale", sprich Kostenvorteile, zu erzielen.

- **Mietvertrag (Service Level Agreements)**

  Das Fundament jeder ASP Beziehung: Hier ist genau festgelegt, welche Leistungen vom ASP zu erbringen sind.

- **Bezahlung nach Zeit und/oder Inanspruchnahme**

  Dies ist ein Teil der SLAs und legt fest, wann und wie die Miete an den ASP zu zahlen ist.

Nahezu als Merkmal von ASP selbst lässt sich in der gegenwärtigen Situation die stark ausgeprägte wirtschaftliche Wachstumsphase darstellen, die ASP momentan durchläuft.

Zu ähnlichen Resultaten kommt das Marktforschungsinstitut IDC [Gillan et al. 1999] , welches sich in seinem Paper jedoch darauf bezieht, was einen "Application Service Provider" ausmacht. Die oben genannte Merkmalsaufzählung stellt aber die Tätigkeit des "Application Service Providings" und die Gemeinsamkeiten aller ASP-Lösungen in den Vordergrund. Diese Betrachtungsweise erlaubt eine viel "objektivere" und grundlegendere Definition von ASP.

1.2. Historie und Abgrenzung von ASP

> *"If you're a CIO with a head for business, you won't be buying computers anymore. You won't buy software either. You'll rent all your resource from a service provider."* Scott McNealy, CEO of Sun Microsystems

ASP ist nicht einfach "erfunden" worden. Vielmehr hat sich ASP aus neuen wirtschaftlichen Bedürfnissen und technischen Möglichkeiten entwickelt.

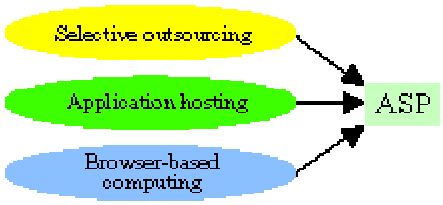

Abbildung 1-2 : "ASP Ingredienzien" [Tao 2000]

Es sind hauptsächlich folgende drei separate Trends, in denen ASP seinen Ursprung findet [Farleit 2000][Tao 2000] :

1.2.1. IT – Outsourcing

IT-Outsourcing ist das Auslagern von IT-Infrastruktur, -Personal, -Prozessen, -Applikationen oder ganzen IT-Abteilungen einer Organisation an einen externen Ressourcenprovider. Meistens werden genau jene Bereiche an andere Organisationen übergeben, die nicht Kerngeschäft des jeweiligen Unternehmens sind (Selective outsourcing).

IT-Outsourcing ist eine Praktik, die seit dem Einzug des PCs in die Büros angewendet wird. In den letzten Jahren (etwa seit dem Jahr 1997 [Berheide 2000]) werden immer spezialisiertere Lösungen angeboten. Angefangen bei grundlegenden Diensten wie Datennetzeinrichtung und -betrieb, über "Messaging"-Lösungen wie E-Mail bis hin zur Übernahme von Softwareapplikationen ("application outsourcing" und "application management"). Letztere Form hat mit ASP so viele Gemeinsamkeiten, dass in der Literatur Uneinigkeit darüber besteht, ob ASP nun eine Unterart von Outsourcing sei oder eine eigenständige Form. Cherry Tree sieht ASP als spezielle Art von "Application Outsourcing" [Cherry Tree 1999], während Farleit Limited [Farleit 2000] IT-Outsourcing als eine von eben diesen drei Einflussgrößen beschreibt, die in der Folge genannt und im Detail beschrieben werden.

In der Regel wird IT-Outsourcing lediglich als Einflussfaktor für die Modelltheorie von ASP gesehen, nicht jedoch für die grundlegende Technik von ASP, nämlich die der "webbasierenden Programme". Dieser Trend wird zuletzt besprochen.
Von allen Trends kann IT-Outsourcing aber in organisatorischer Hinsicht den größten Einfluss verzeichnen. Folgende Vergleichstabelle [Steffen 1999] dürfte Klarheit über die organisatorischen Unterschiede von IT-Outsourcing und ASP verschaffen und zeigt deutlich, dass ASP sich sehr viel konkreter manifestiert als IT-Outsourcing:

	Unternehmensinterne Informations-verarbeitung	**Traditionelles Outsourcing**	**Application Service Providing (ASP)**
Infrastruktur	Der Anwender kauft, installiert und finanziert die Hardware und Systemsoftware	Sowohl der Anwender als auch der Anbieter können Eigentümer der Infrastruktur sein	Der Anbieter besitzt und betreibt die Hardware
Standort der Informationstechnik	Im Hause des Anwenders	Bei Anbieter oder Anwender	Beim Anbieter
Administration der Informations-verarbeitung	Der Anwender verwaltet das System	Der Anbieter verwaltet das System beim Anwender oder extern	Der Anbieter verwaltet das System extern
Softwarekosten	Der Anwender besitzt die Lizenzen	Der Anbieter verkauft oder vermietet die Lizenz an den Anwender	Der Anbieter besitzt und verwaltet immer die Lizenzen
System-implementierungs-kosten	Der Anwender passt die Lösungen auf eigene Kosten an seine Geschäftsprozesse an	Der Anbieter passt die Software kostenpflichtig an die Geschäfts-prozesse des Kunden an	Die Implementierungs-kosten fließen in die variable Leistungs-abrechnung ein
Bindung an den Partner	Langfristig	Mittel- bis langfristig	Kurz- bis mittelfristig
Individualität der Lösung	Sehr stark ausgeprägt	Stark ausgeprägt	Gering ausgeprägt

Tabelle 1 : Vergleich von IT-Betrieb [Steffen 1999]

Die größte Schnittmenge von ASP und Application Outsourcing bildet das Geschäfts- und Vertragsmodell. Einige Aspekte im Kapitel "Organisation und Technik" werden daher auch allgemein auf IT-Outsourcing zutreffen.

Wenn man IT-Outsourcing jedoch als eine Modellgrundlage von ASP sieht, und ASP als einen speziellen Anwendungsfall von IT-Outsourcing, so lässt sich die Schwerpunktsetzung dieser Arbeit leicht abgrenzen, und es macht Sinn auf IT-Outsourcing nicht näher einzugehen. Daher soll die Praxis des IT-Outsourcings an dieser Stelle nicht näher betrachtet werden und auf die Literatur verwiesen werden. Eine umfangreiche Abhandlung des Themas, die auf der Fachhochschule Zentralschweiz entstanden ist, ist unter [FHZ 2000] zu finden.

1.2.2. Application hosting

Seit den Anfängen des Internets ermöglichen die Internet Service Provider (ISPs) sowohl für Private als auch für Großkunden den Zugang zu diesem Medium. Die Zahl der angebotenen Dienste und die Produktvielfalt wachsen ständig: Als das Angebot über die Basisdienste wie Internetzugang, News oder E-Mail hinausgegangen war, und durch die Übernahme und Betreuung von Web-Sites durch den ISP die Speicherung von personalisierten Daten beim ISP

immer intensiver betrieben wurde, tauchte für diese neue ISP-Kompetenz der Begriff "Hosting" auf (sogenannte Hosting Service Providers (HSPs)). Hosting bedeutet Speicherung und zur Verfügung stellen von Fremd-Daten und -Applikationen. Im allgemeinen ist der HSP nur der Auftragnehmer für, in diesem Beispiel, Application Hosting. Die Softwarelizenz gehört nicht dem HSP.

Im Gegensatz dazu betreibt ein ASP selbst internetbasierte Software, dessen Lizenz dem ASP gehört, oder lagert nur den Betrieb bspw. an einen eben solchen HSP aus. Üblicherweise liegt die Verantwortung für den angebotenen Inhalt nicht beim HSP, sondern beim Anbieter, wobei aber Mischformen immer üblicher werden, sodass viele Application Hosting Firmen immer öfter als Application Service Provider auftreten. Diese Möglichkeit können viele ISPs auch dadurch wahrnehmen, dass sie große Erfahrungen in den Bereichen Internet, Server und Serverapplikationen mitbringen.

1.2.3. Browser-based computing

Der dritte Trend bzw. ASP-Aspekt bestimmt die Technik von Application Service Providing: Internetseiten haben sich von reinen Informationsseiten mit statischem Inhalt, wie Texte und Bilder, zu dynamischen, interaktiven "Portalen" entwickelt. Websprachen wie JavaScript und Java erhöhen die Funktionalität. Serverseitige Internetsprachen wie Java-Servlets oder Microsoft Active Server Pages ermöglichen einen individuellen Informationsgehalt, je nach Benutzer und Anforderung. Mehr Dynamik und Individualität in den Internetseiten ist das Resultat.

Die entscheidende Entwicklung hin zum Application Service Providing machte das sog. "Portal computing" durch das Anbieten von Anwendungsdiensten, das heißt, dem Benutzer mehr als bloße Information zur Verfügung zu stellen. Konzeptuell stehen für solche Angebote im Vordergrund: der praktische Nutzen für den Anwender (die Anwendung), das Operieren auf Benutzereingaben durch Internetprogramme (Funktionalität) und die Lauffähigkeit der Programme in einem Internetbrowser (Standardisierung). Auf diese technischen und praktischen Aspekte von ASP wird in dem Unterkapitel "Technologiesicht" noch eingegangen.

Internetbasierende Dienste und Programme, insbesondere dann, wenn sie kommerzialisiert werden, weil der praktische Nutzen für den Anwender groß genug ist, werden in zunehmendem Ausmaß ununterscheidbar von dem, was ASPs anbieten.

Obwohl die eben besprochenen Trends von unterschiedlichem Ursprung sind, beeinflussen sie doch stark die eingangs erwähnten Merkmale von ASP.

Zusammenfassend kann man sagen: IT-Outsourcing nahm Einfluss auf das (Geschäfts-) Modell von ASP, Application Hosting lieferte die Grundlage für die Verbreitung von ASP und Portal Computing lieferte das Fundament für die technische Grundlage von ASP.

1.3. Sichtweisen von ASP

Im letzten Kapitel wurden ausführlich Einflussgrößen und Ideen zum ASP Modell beschrieben, aber noch nicht die Motivationen, Chancen und Risiken für ein solches aufgezeigt. Dies soll in diesem Kapitel durch drei Sichtweisen geschehen: Durch eine Analyse seitens des Anwenders ("Anwendersicht"), seitens des Anbieters ("ASP - Sicht") und seitens des technischen Prinzips von ASP ("Technologiesicht").

1.3.1. Anwendersicht

Die Anwender, sprich die Endkunden, sollen am meisten von ASP profitieren. Als Referenz-Anwender für die folgende Analyse sollen im folgenden kleine und mittlere Unternehmen ("KMUs", "SMEs" (Small and Medium Enterprises)) dienen. Sie stellen die Hauptzielgruppe von ASP dar (siehe Kapitel "Applikationen und Adressaten" in dieser Einführung). 66% der derzeitigen Abnehmer von ASP-Lösungen sind mittelständische Firmen [Berheide 2000].

Bei der Aufzählung von Vor- und Nachteilen von ASP für den Anwender stellt sich die Frage, wie weit Theorie und Praxis auseinander klaffen. Es sind hauptsächlich Aspekte technischer Natur, die die ASP Praxis von dem ASP-"Hype" unterscheiden. Andererseits ist eine ständige Weiterentwicklung zu beobachten. Somit wäre eine plakative Darstellung mittels Fakten im einzelnen nicht immer zutreffend. Die beschriebenen Aspekte sind daher allgemeiner Art, im Vergleich mit "traditioneller" Softwarebeschaffung und -nutzung und zum gegenwärtigen Zeitpunkt zu sehen.

Unter diesen Bedingungen ergeben sich aus Anwendersicht folgende Vorteile (im Anschluss daran werden mögliche Nachteile dargestellt):

- **Zeit-, Kosten- und Personalreduktion**

  Mit der zentralen Speicherung, Verwaltung und Echtzeitbereitstellung von Applikationen und Daten durch den ASP entfallen zeitaufwendige Installations-, Update- und Wartungsarbeiten (wie zum Beispiel Backups).

  Mit Beginn des Mietvertrages schaltet der ASP die gewünschte vorkonfigurierte Software frei. Für manche Lösungen wird ein kleines Plug-In für den Browser oder ein eigenes proprietäres Clientprogramm benötigt (näheres zu technischen Details im Kapitel "Organisation und Technik"), das aber ebenfalls online bezogen werden kann. Der Kauf und die Aufbewahrung von Installations-CDs entfällt also.

  Softwarekonflikte mit anderer lokal installierter Software werden seltener, da die Datenverarbeitung zum großen Teil am Server stattfindet und der restliche Teil nur innerhalb der Browserumgebung abläuft. Sofern diese sog. "Sandbox" im Browser auch "Bug-free" ist, lässt sich damit eine Erhöhung des Schutzes vor Viren feststellen. Die Verantwortung für einen aktuellen Virenschutz liegt beim Provider. Client-Rechner und Server-Rechner sind in hohem Maße bei Angriffen auf Daten entkoppelt.

  Software-Updates finden termingerecht und gleichzeitig bei allen Mitarbeitern statt, sodass Inkompatibilitäten durch verschiedene Programmversionen auf unterschiedlichen Rechnern seltener vorkommen. Daten werden vom Provider gesichert, teure und zeitaufwendige "In-House"-Sicherungsoperationen entfallen. Eigenes Personal, das für solche Prozeduren benötigt werden würde, kann eingespart werden. Aufgrund der Skalierbarkeit von ASP-Anwendungen muss der Provider Verwaltungsoperationen (wie Backups, Virenschutz, Updates, etc.) nur pro Applikation, Betriebssystem oder Server und nicht pro Benutzer durchzuführen, und diese Zeit- und Arbeitsersparnisse können an den Endkunden in Form von Kostenreduktionen (gegenüber herkömmlicher Software) weitergegeben werden.

  Hinzu kommt, dass sich KMUs komplexe, nicht ASP-basierte Software einfach nicht leisten können. Ein ASP kann durch den Online-Vorteil auf einfache Weise wesentlich differenziertere, für jeden Bedarf geeignete Software anbieten. Dadurch wird die Attraktivität von Software größer, was die Absatzzahlen erhöht und letztendlich den Preis für den Endkunden verringert. Ein gutes Beispiel für einen solchen Sachverhalt stellt "R/3" von SAP dar ("R/3" von SAP ist ein komplexes Softwarepaket zur Automatisierung von Geschäftsabläufen [SAP 1999]). Durch die Nutzung der ASP-basierten Lösung

"mySAP.com“, anstatt das komplette Softwarepaket zu kaufen, kann ein KMU wesentliche Kostenvorteile erzielen.

Nicht nur Kostenvorteile bei Software, sondern auch bei Hardware stehen bei Application Service Providing im Vordergrund: Das Grundsystem von heutiger Software stellt immer größere Hardwareanforderungen, während die Mehrbenutzerfähigkeit und -performance von Betriebssystemen und Anwendungen immer größer werden. Seit dem Internet nimmt auch die selektierte Informationsbeschaffung einen neuen Stellenwert ein, sodass Datenbankanbindungen zu einem integralen Bestandteil von Applikationen werden. Ein Trend, der dem Application Service Providing sehr zugute kommt: Wo früher mindestens ein Pentium 200MHZ vonnöten war, um komplizierte Geschäftsprozesse abzuwickeln, ist mit ASP unter Umständen schon ein 486-66MHZ (genauere Untersuchungen in Kapitel 3) als Client ausreichend, da der Client-PC, wie bereits erwähnt, nur Rendering- und Benutzereingabefunktionalitäten zur Verfügung stellen muss. Datenbankanbindung und Pflege des Dateisystems sind ausschließlich Aufgaben des Servers bzw. des Providers. Die Frage nach Performance (und der damit verbundene Zeit- und Kostenaufwand für Aufrüstung u.ä.) wird damit nahezu vollständig dem Provider überlassen. Außerdem bieten manche ASPs auch Komplettpakete inklusive Internetzugang, -hardware (wie zum Beispiel spezielle, für ASP optimierte Client-PCs (sog. Network-Computer) oder Router), -konfigurierung und -wartung an, sodass für den Kunden auch in diesen Bereichen weniger an Hardware und Know-how zu investieren ist.

Einer Analyse von "Stratecast Partners" zufolge, belaufen sich die Kostenreduktionen für eine Firma mit 10 PCs zwischen 28% und 47% [c-quential 2000].

- **Statt IT – Investitionen: Variable Kosten**

Durch das Mietmodell bedingt werden neue Abrechnungsmodelle geschaffen: Während für "packaged software" zum Erwerb oder zum Betrieb eine einmalige Lizenzgebühr zuzüglich eventuelle weitere Gebühren für Updates erforderlich sind, werden die Nutzungsgebühren bei ASP zeit-, benutzer- oder transaktionsgesteuert eingehoben. Das heißt im einzelnen:

Der Benutzer zahlt nur für die Zeit, in der er eine Applikation auch tatsächlich benützt (d.h. geöffnet hat), für jeden definierten Aufruf von Programmfunktionen, eine monatliche Flatrate oder eine Kombination aus den genannten Kosten. Flatrates werden pro Benutzer oder pro Rechner (genaugenommen pro Installation eines Client-Plug-Ins) kalkuliert und haben den Vorteil, die ASP-Kosten besser abschätzen zu können.

Die angebotenen Dienste und Leistungen können dabei genauso wie die Preise in einem weiten Spektrum variieren: Zum Beispiel verschiedene Sicherheitsstufen (Sicherheit vor Datenverlusten und -manipulation), Grad der Vertraulichkeit der Daten oder Umfang, Verfügbarkeit und Performance der Server-Ressourcen. Der Provider kann vielschichtige Angebote für jede Zielgruppe anbieten, aus denen der Kunde wählen kann. Alle Leistungen und Verpflichtungen seitens des Providers und des Kunden werden in den Service Level Agreements festgehalten, die in Kapitel 1.7.2. beschrieben sind.

Durch ASP wird es kleineren Firmen mit wenig Startkapital wesentlich leichter gemacht, eine EDV mit professioneller Software aufzubauen und somit auch gleich mit größeren Firmen zu konkurrieren, da große Vorabinvestitionen im Bereich Software (und teilweise Hardware) wegfallen.

Eine Kosten-/Nutzenrechnung von ASP auf Basis von Windows 2000 Advanced Server (RDP 5.0) mit anschaulichen Grafiken ist in [grouptech 2002] sowie im Kapitel "Wirtschaftliche Aspekte" zu finden.

- **Bessere Produktvergleichbarkeit**

  Das Angebot im Internet erleichtert wesentlich die Suche nach dem "besten" Programm. Zu fast jedem Programm auf ASP-Basis werden Demoversionen angeboten, die nicht wie sonst gewohnt heruntergeladen werden müssen, sondern gleich online ausprobiert werden können. Es kann nicht zu Softwarekonflikten oder Virenübertragungen kommen, da kein Programmcode am Client-PC installiert werden muss. Das Einholen von Informationen und das Testen von Softwareprodukten vor dem Kauf gehen mit dem Internet Hand in Hand. Ein Provider kann auch Softwareprodukte, die untereinander konkurrenzieren, und den gewünschten Support in seinem Programm anbieten, während der bestmögliche Support für herkömmliche Software meist nur vom Hersteller oder autorisierten Händler gewährleistet ist.

  Auf http://www.runaware.com kann man online eine breite Palette von Applikationen gratis ausprobieren.

- **Geräte- und Ortsunabhängigkeit**

  Zwei der wichtigsten Ziele von ASP überhaupt: Die Hardwareentwicklungen in den letzten Jahren zeigen deutlich, dass die Zukunft im Bereich der mobilen Kommunikation und Softwarenutzung liegt. Speziell die Internetfähigkeit der sogenannten "Palm-Tops" wird sich in den nächsten Jahren entscheidend verbessern, die Leistung eines Desktop-PCs wird jedoch vergleichsweise nur langsam erreicht werden können. Massenspeicher wie CD-ROM Laufwerke können aufgrund ihrer Größe, hohe Prozessortaktfrequenzen aufgrund großen Stromverbrauchs nicht für Handheld-PCs verwendet werden. ASP ist daher für solche Geräte prädestiniert, da es kein CD-ROM Laufwerk und auch keinen starken Prozessor von der clientseitigen Hardware abverlangt. Dieses Konzept eines minimalisierten und auf Internet spezialisierten PCs wird mit "Thin-Client" oder "Network Computer" bezeichnet.

  ASP ermöglicht eine Nutzung von Software mit jedem Gerät, das einen Internetbrowser implementiert. Der Anwender bestimmt, ob er dieselbe Software von seinem Heim-PC, von dem Firmen-Laptop oder von seinem Palmtop aus benutzt.

  Im Idealfall könnte der Server sämtliche Browsersprachen wie zum Beispiel HTML, JavaScript, Java oder WAP unterstützen und dem Benutzer einen auf die Fähigkeiten des Client-Gerätes abgestimmten Kontext anbieten.

  ASP ermöglicht eine Geräteunabhängigkeit in software- und hardwaretechnischer Hinsicht: Applikationen und Daten stehen zu jeder Zeit auf jedem internetfähigen Gerät zur Verfügung. Diese Geräte können von einem beliebigen Hersteller stammen und mit beliebigem Betriebssystem und Prozessor ausgerüstet sein, solange ein Internetbrowser verfügbar ist. Firmenmitarbeiter können, wenn sie sich außerhalb ihres Büros befinden (z.B. zu Hause, bei Firmenpartnern, bei Kunden oder in Hotels) mit ihren Applikationen arbeiten, ohne ihren Laptop mitnehmen zu müssen, solange ein internetfähiges Gerät zur Verfügung steht.

  Teamarbeit, über der ganzen Welt verstreut, wird wesentlich vereinfacht, vorausgesetzt die Arbeitsschritte der einzelnen Mitarbeiter sind für jedes Teammitglied nachvollziehbar, da eine explizite Datenübertragung und -synchronisation nicht notwendig ist.

  Es können Anwendungen entworfen werden, die aktuelle externe Daten (wie z.B. Börsenkurse) automatisiert über das Internet beziehen, ohne dass der Benutzer sein Endgerät eingeschaltet haben muss, da der Provider für einen durchgängigen Tag- und Nachtbetrieb seiner Server und Applikationen sorgt.

  Die Geräte- und Ortsunabhängigkeit, die durch ASP geschaffen wird, ermöglicht Geschäftsleuten mobiler zu arbeiten, besser und schneller informiert zu sein und schneller zu reagieren.

- **Hohe Technologiekompetenz des Dienstleisters**

  Das ASP-Modell verlangt vom Provider eine hohe Kompetenz in den Bereichen Internet, Server und Serverapplikationen: Nicht nur, dass mehrere Kunden mit den Applikationen zu beliefern sind, es muss, wie in den Service Level Agreements vereinbart, auch eine gewisse Mindestqualität sichergestellt werden. Zudem steigen besonders ISPs und HSPs in das ASP-Geschäft ein, die solche Erfahrungen schon mitbringen. Auf der anderen Seite erwarten sich ASP-Kunden mindestens dieselbe Qualität und ähnliche Preise in der Softwarenutzung, wie sie es von ihrer herkömmlichen Software gewohnt waren.

  Ein aktuelles Thema, wie es in [Berheide 2000] zitiert wird, ist der Mangel an qualifiziertem IT-Fachpersonal: Besonders Unternehmen, die nicht aus dem IT-Bereich stammen und IT-Support ab und zu auf Abruf benötigen, werden mit dem ASP-Modell billiger und besser mit IT-Fachwissen versorgt. Auf diese Art kommt die Expertise des ASP-Fachpersonals durch die zentrale Verwaltung gleich mehreren Kunden gleichzeitig zugute.

- **Bessere Marktchancen**

  Die eben genannten Vorteile lassen sich unter diesem Punkt zusammenfassen: Durch ASP ist es möglich, günstiger und schneller als bisher, neue und kostspielige und/oder komplexe Software in Betrieb zu nehmen.

Die eben genannten Vorteile sind Möglichkeiten, die das ASP Modell bietet. In der gegenwärtigen Praxis, ohne in dieser Einführung auf die einzelnen Angebote und Lösungen näher einzugehen, müssen jedoch auch folgende Restriktionen oder Risiken in Kauf genommen werden:

- **Kostenreduktionen nicht immer eindeutig**

  Bevor von einer Kostenreduktion die Rede sein kann, müssen umfangreiche Analysen betrieben werden: Über welche Zeit hinweg und wie häufig sollen welche Anwendungen genutzt werden ? Mit welcher Performance und auf welcher Sicherheitsstufe ?

  Bekannt ist, dass sich die Kosten einer IT-Lösung aus den Blöcken Softwarelizenzen, Implementierung, Infrastruktur sowie Service und Wartung zusammensetzen. Einer Studie der "MetaGroup" zufolge (wird zitiert in [Igler 1999]) ergeben sich positive Effekte hinsichtlich der IT-Kosten vor allem beim kurzfristigen ASP-Einsatz (0-2 Jahre lang). Die großen Kostenblöcke einer Software-Lösung (Implementation, Service und Wartung) wirkten sich langfristig (4-6 Jahre lang) sogar negativ aus.

  Dies sei dadurch zu erklären, dass die Implementierungskosten (Reengineering der Geschäftsprozesse, Anpassungen der Software ("Customizing"), Migration von Daten und Schulungen) in etwa gleich blieben, da das ASP Modell hierbei keinerlei Synergieeffekte beisteuern könne. Nur branchenspezifisch vorkonfigurierte Software könne hier eine Kosteneinsparung von 10-20% erbringen.

  Bei der IT-Infrastruktur muss man ebenfalls aufpassen: Es gibt Branchen, für die ein Internetzugang wenig sinnvoll oder gewinnbringend ist und die daher bisher ohne Internetzugang gearbeitet haben. Für ASP ist ein Internetzugang (mindestens ISDN-Geschwindigkeit) jedoch Grundvoraussetzung. ASP-Softwarebenutzung als einzigen gewinnbringenden Teil des Internetzugangs rentiert nicht. Es ist daher genau zu prüfen, ob trotz der zusätzlichen Internetnutzungs- und -anschaffungsgebühren (egal ob sie nun beim ISP oder beim ASP entrichtet werden) noch von einer Kostenreduktion die Rede sein kann.

Nur wenn ein Internetzugang integraler Bestandteil des Geschäftsbetriebes ist, kann sich ASP überhaupt anbieten.

Weniger komplexe oder unternehmenskritische Applikationen wie z.B. Office oder Spiele über einen ASP zu nutzen macht ebenfalls wenig Sinn, weil es in diesen Kategorien herkömmliche, preisgünstige und kostenlose Softwarealternativen gibt. Besonders fällt dies dann ins Gewicht, wenn durch ASP zusätzliche Miet- und Internetgebühren fällig werden, die die Softwarelizenzgebühren solcher Software nach kurzer Zeit übersteigen.

- **Starke Abhängigkeit von ASP und Internet**

Durch den Service Vertrag mit dem ASP und die ständige Internetanbindung bestehen starke Abhängigkeiten zwischen ASP, Internet und Endkunde. Im Falle eines Teil- oder Ganzausfalles einer ASP-Applikation ist in den SLAs definiert, wie weit und in welchem Ausmaß der ASP haftet. Im "Basic-Service-Level" wird lediglich die monatliche Grundgebühr zu definierten Teilen erlassen [c-quential 2000]. Dies setzt zwar den ASP unter Druck, nützt dem Kunden aber wenig, von einem Schadensersatzanspruch für entgangene Geschäfte kann nicht die Rede sein. Dieser wird nur in den "Extended Services" zu einer höheren Grundgebühr angeboten. In Kapitel 1.4.4., "Arten von Application Service Providern", wird darauf noch genauer eingegangen.

Zudem erlauben ASP-Angebote, per Definition, nur einen geringen Grad an Anpassung durch einen Kunden. Die große Flexibilität eines PCs wird durch ASP zunichte gemacht, da die Browserumgebung nur eine geringgradige "Customization" erlaubt. Durch die zentralistische one-to-many Philosophie von ASP wird die Einflussnahme vom Kunden auf Programmeinstellungen und Ressourcen, wie z.B. das Dateiablagesystem, Speicherplatz, Performance, Utilities, Patches für die Applikationen usw., bewusst restriktiv gehalten, damit eine missbräuchliche oder versehentliche Fehlbedienung nicht zum Kollaps des gesamten Applikationsservers führt.

Sofern es in den SLAs so definiert wurde, werden Programmupdates vom ASP und nicht vom Kunden durchgeführt. Es entsteht auf diese Weise eine große Abhängigkeit, was Hard- und Softwareänderungen betrifft.

Wenn ein ASP seine Arbeit z.B. aus wirtschaftlichen Gründen einstellen muss, kann das ganz erhebliche Auswirkungen, so wie sie den SLAs definiert sind, auf seine Kunden haben. Übliche Vorgehensweisen in einem derartigen Fall werden in Kapitel 1.7.3. diskutiert.

Eine zweite Unsicherheit betrifft das Internet: Weder in vertraglicher noch in technischer Hinsicht ist eine Übertragung mit definierten Mindestantwortzeiten garantiert. ISPs übernehmen nicht die volle Verantwortung für Netzausfälle. Denkbar ist auch, dass sich die Beweisführung im Falle eines Netz- oder Applikationsausfalles schwierig gestaltet. Es wurde sogar schon ein Paper mit Richtlinien zur Streitschlichtung veröffentlicht [ASP I.C. 2000].

Bei einer Umfrage (Forrester 1999, zitiert in [Gruber 2000]) von IT-Unternehmen nach den Gründen, warum sie gegenüber ASP-Lösungen noch mit Zurückhaltung entgegnen, antworteten 71% sinngemäß: "Wegen Verlust von Kontrolle".

Technische Ausfälle können zwar bei jeder Art von Softwarenutzung passieren, aber bei ASP ist die Einflussnahme des Anwenders auf den Server- und Softwarebetrieb sehr eingeschränkt.

- **Meist hohe Kundenbindung**

Provider versuchen aus betriebswirtschaftlichen Gründen, ihre Kunden stark an ihr Produkt zu binden. Softwaremietverträge belaufen sich üblicherweise auf eineinhalb bis drei Jahre [Cherry Tree 1999]. Im Vertrag ist festgelegt, was mit Applikationen und Daten im Falle

einer (frühzeitigen) Auflösung des Vertrages geschieht. Technische Schwierigkeiten bei der Migration der Daten (zu einem anderen Service-Provider oder zurück zu einer In-House-Lösung) erhöhen die Kundenbindung zusätzlich.

- **Datenschutz- und Datensicherheitsprobleme**

  Bevor ein Vertrag mit einem ASP eingegangen wird, stellt sich für Unternehmen die Frage, ob ausreichend Vertrauen besteht, vertrauliche und unternehmenskritische Daten an eine Drittfirma wie einen ASP zu übergeben [Tao 2000][Arthur D Little 2000][Franssen 2000]. Bereits bei Unterzeichnung des Vertrages werden Firmen- und Personaldaten aufgenommen. Während des Betriebes sind für den ASP alle Geschäftsvorgänge aller seiner (untereinander konkurrierenden) Kunden sichtbar. Der ASP unterliegt absoluter Verschwiegenheitspflicht der Kundendaten. Durch fehlende Kontrollmöglichkeit und geringe Einflussnahme von Seiten des Kunden auf ASP-Datenschutzaspekte verkauft sich unternehmenskritische Software (z.B. ERP-Systeme) noch relativ schlecht über einen ASP [Tramm 2000].

  Ein zweiter Unsicherheitsfaktor bestimmt die Datensicherheit: Die bekannten Angriffe auf Daten ("abhören", verändern, einschleusen), absichtlich oder unabsichtlich, durch Mitarbeiter des Providers, durch Viren oder externe Hacker müssen vom ASP durch strengste Sicherheitsprozeduren verhindert werden. Die modernen Verfahren zur Verschlüsselung (in erster Linie VPN über SSL) gelten zwar aus heutiger Sicht als sicher, doch bleiben grundsätzliche, zusätzliche Sicherheitsprobleme bei ASP bestehen: Zum Beispiel durch die Orts- und Geräteunabhängigkeit von ASP kann ein Hacker von jedem Gerät mit Internetbrowser durch bloße Eingabe oder Knacken des Benutzernamens und Passwortes auf Applikationen und Unternehmensdaten zugreifen. Daher müssen auch auf Clientseite noch umfangreiche Autorisierungsmaßnahmen und Zugriffsbeschränkungen entwickelt und angewendet werden. Im Unterkapitel "Sicherheit" von Kapitel 2.6. werden Sicherheitsanforderungen an einen ASP erläutert.

- **Kaum standardisiert**

  Gegenwärtig werden die Standards für webbasierte Programme erst entwickelt. Diese Standards werden notwendig, um die Probleme der "Service Integration" zu lösen: Es wurden mit HTML, Java, JavaScript, Windows Terminal Services auf Basis von Remote Desktop Protocol (RDP), Citrix MetaFrame auf Basis von Independent Computing Architecture (ICA) zwar Standards und Quasi-Standards gesetzt, die aber noch nicht von allen Geräten und Browsern unterstützt werden.

  Sowohl für Client- als auch für Server-Front-Ends müssen erst Möglichkeiten geschaffen werden, standardisiert mit Fremdapplikationen (serverseitig zum Beispiel mit Software zur Abrechnung, clientseitig zum Beispiel mit einem lokal installierten E-Mailprogramm) zu kommunizieren. Der Zugriff oder die Synchronisation mit lokal installierter Software oder von einem anderen ASP auf Daten, die beim ASP liegen, wird erschwert, wenn man sich nicht auf die Nutzung der serverseitig installierten Applikationen beschränkt.

- **Browser oder Browser-Plug-In ist kleinster gemeinsamer Nenner ([SCN Education 2000], S. 23)**

  Um die Geräte- und Ortsunabhängigkeit von ASP zu gewährleisten, ist man bemüht, am Client Rechner lediglich einen Internetbrowser vorauszusetzen. Während man mit HTML, Java und JavaScript-Technologien derzeit erhebliche Einschränkungen in Kauf nehmen muss (absturzanfällig, inkompatibel zu unterschiedlichen Browsern, eingeschränkte

Funktionalität, was z.B. Fenster, Menus oder den Zugriff auf das Server-Dateisystem betrifft), erlauben die Citrix MetaFrame-, Windows Terminal Service- und SCO Tarantella-Technologien, auf eine virtuelle Betriebssystemsession des Servers (z.B. Windows oder UNIX) zuzugreifen. Der Nachteil ist, dass der Benutzer an die serverseitige Betriebssystemsplattform gebunden ist und dass das clientseitige Betriebssystem in den Hintergrund gedrängt wird. Mit letztgenannten Technologien ist derzeit außerdem eine Integration von Fremdapplikationen und -daten unmöglich.

Es scheint, als wäre eine optimale, reibungslos integrierbare und standardisierte Art der Softwarenutzung per Internetbrowser noch nicht verfügbar.
Derzeit verfügbare Techniken werden im Kapitel 3 vorgestellt.

- **Markt noch nicht reif**

Einer Studie von A. Apfel und Gartner zufolge (zitiert in [Stahlknecht 2000]), werden im Jahr 2001 60% aller derzeitigen tätigen ASPs ihren Dienst eingestellt haben. Im Jahr 2000 gab es etwa 480 ASPs weltweit [Arthur D Little 2000]. In den USA nutzen zwar bereits 80% aller Unternehmen einen ASP, die Marktdurchdringung in Europa ist jedoch noch weit davon entfernt. Standards, Soft- und Hardwareprodukte und ASP-Angebote müssen noch einen Reifeprozess durchmachen, bis es zur Konsolidierung des ASP-Marktes kommt.

Es macht Sinn, einen Blick auf den österreichischen Markt zu werfen, da aufgrund der Vertragsmodalitäten und der Vorteile der geographischen Nähe für ASP-Kunden in Österreich nur inländische ASPs in Frage kommen. In Österreich wächst der Markt nur langsam [IC 2001]. Dieser ist stark von der Nachfrage getrieben: Erst 4% der Unternehmen nutzen einen ASP. Demgegenüber stehen 28% Interessenten. Viele Vorteile von ASP müssen erst kommuniziert werden. ASP scheitert hierzulande auch oft an den technischen Voraussetzungen (insbesondere am raschen Internetzugang).

In Österreich gibt es mehr Application Infrastructure Provider (AIPs) als ASPs, es sind einige AIPs dabei, die noch gar keinen ASP als Kunden haben. Die folgende Liste an AIPs und ASPs in Österreich ist beinahe vollständig (in Klammern das Angebot an Anwendungen). (Die Softwarehersteller wurden ausgespart, da hierbei deren Standort unerheblich ist):

- **Österreichische AIPs**: KPNQWest, Nextra, VIANET, Siemens Business Services, Unisys, Datakom, PSINet
- **Österreichische ASPs: AIinformatics** (ERP-Software "mySAP"), **factordatasystem's** (Friseurverwaltung "StyleVIPNet"), **Intra-Sys** (verschiedene Windows-Software auf Citrix-Basis), **Schrack** (verschiedene Officeapplikationen, E-Commerce und Groupware), **Informatec ASP** (gemischtes Angebot: Sun Star Office, SAP, Cross TV), **Netway** (ASP Pilotprojekt für Trafiken)

Der Markt in Österreich ist definitiv noch ausbaufähig.

1.3.2. ASP – Sicht

Marktanalysten sprechen dem ASP-Modell erhebliche Marktchancen zu (Marktdaten hierzu in Kapitel 1.6.). Daher drängen IT-Unternehmen, unabhängig von ihren Erfahrungen auf diesem Gebiet, in den ASP-Bereich hinein. Unternehmen müssen sich technischen Umstellungen unterziehen, Know-how akquirieren, Partnerschaften eingehen und neue Geschäftsmodelle entwickeln. Unternehmen, die in das ASP-Geschäft einsteigen, werden daher mit folgenden Chancen und Risiken konfrontiert:

- **Neuausrichtung von Unternehmen und starker Wettbewerb**

  ASP erfordert Fachwissen und Produkte aus nahezu jedem IT-Bereich: IDC [Gillan et al. 1999] nennt die Software-, die Hardware-, die Service- und die Kommunikationsindustrie. Alle diese ASP-Player (ob sie nun selbst ASPs werden oder mit einem ASP kooperieren) werden in diesem neuen Markt mit Fragen konfrontiert: Wer wird mein Kunde sein? Mit wem werden wir Partnerschaften abschließen? Wie wird ASP unsere Preis-, Verkaufs- und Marketingstrategie beeinflussen? Welche Möglichkeiten und Bedrohungen sind zu erwarten?

  Diese Fragen zu diskutieren ist Aufgabe des strategischen Informationsmanagements. In kleinen und mittleren Unternehmen obliegt diese Aufgabe der Geschäftsführung (Schlussfolgerung von [Stahlknecht 2000]).

  Durch die großen Erwartungen, die in ASP gesteckt werden, und die prognostizierten Erfolgsaussichten begründet, kommt es zu starkem Wettbewerb, Firmenpartnerschaften und -zusammenschlüssen. Weniger erfolgreiche ASP-Anbieter könnten jedoch, wie bereits erwähnt, zur Einstellung ihres ASP-Angebotes gezwungen werden.

  ASP-Player werden im nächsten Kapitel vorgestellt, optimale Erfolgsstrategien für ASPs herauszufinden ist jedoch nicht Teil dieser Arbeit. In [Gillan et al. 1999] und [Tramm 2000] werden ASP-Einflussgrößen und Strategien erläutert.

- **Partner- und Kundenorientierung**

  Ein ASP schreibt nur im Ausnahmefall die Software für seine ASP-Lösungen selbst [Farleit 2000]. Statt dessen kooperiert er mit zahlreichen Experten wie ISPs, Service Integratoren, Hard- und Softwareanbietern und -distributoren und Wiederverkäufern. Kooperationen sind für den ASP sowohl Chance als auch Risiko, da der ASP als Generalunternehmer gegenüber dem Kunden Alleinverantwortlichkeit trägt. Eine reibungslose Zusammenarbeit mit auf das ASP-Lösungsportfolio spezialisierten, kompetenten Partnern erweist sich daher als Schlüsselfaktor für den Erfolg.

  Die grundsätzliche Frage in diesem Zusammenhang ist, ob man als Unternehmen selbst ASP wird oder mit einem ASP kooperiert und wer die Kunden dann sein werden. Besonders für die Softwareproduzenten ergibt sich eine Verschiebung von der Massenproduktion für Endkunden weg hin zu einigen wenigen Software-Lizenzen für ASPs.

  Einen Einfluss auf das Kundeninteresse für einen ASP-Anbieter kann auch die Reputation der Partnerfirmen haben [Arthur D Little 2000].

- **Branchenwissen**

  Der Markt lässt sich in Anbieter für horizontale und vertikale Lösungen einteilen. Horizontale Anbieter vertreiben eine Art von Software, bspw. ein Enterprise Ressource Planning Programm oder eine Customer Relation Management Anwendung branchenübergreifend. Sie sind auf Spezifikationen der Software spezialisiert und haben in branchenspezifischen Prozessen nur sehr wenige Erfahrungen. Vertikale Anbieter auf der anderen Seite sind auf bestimmte Branchen fixiert und bieten für diese Unternehmen spezielle oder speziell angepasste Software an [Berheide 2000]. Vertikale Anwendungen müssen in der Regel öfter aktualisiert werden und erfordern einen höheren Kunden-Support (z.B. Hotlines) als horizontale Anwendungen. Damit sind sie prädestiniert für ASP, denn diese Dienstleistungen lassen sich leicht in das Servicemodell von ASP integrieren.

  ASP-Kunden von vertikaler Software sind jedoch sehr bedacht darauf, eine gewisse Menge an Branchenwissen zu fordern, insbesondere wenn unternehmenskritische Prozesse und Daten an den ASP übergeben werden. Schließlich bestimmt die Art und das Konzept

der Software(-nutzung) den Unternehmenserfolg des Kunden. Der Kundensupport sollte über die technische Auskunft hinausgehen, der ASP sollte einziger Ansprechpartner für alle Fragen und Anliegen zur ASP Lösung sein.

TDS [Igler 1999] führt auf, dass Branchenwissen gar von 74% aller Befragten gefordert wird.

- **Branding**

Ein Nebenaspekt beim Konzipieren eines ASP-Angebotes ist die Markenzugehörigkeit und Namensgebung. Ein herkömmliches Softwareprodukt von Firma X wird von Firma Y ASP-tauglich gemacht und vom ASP Z angeboten. Aus dem ursprünglichen Produkt des Softwareherstellers wird somit ein Produkt des Applikation Service Providers, der ja auch den kompletten Support anbietet. Urheberrechte, Namensgebung, Markenzugehörigkeit und Kundenbetreuung sind wesentliche Diskussionspunkte für die Zusammenarbeit zwischen einem ASP und seinen Partnern.

- **Verantwortung**

Ein Wartungsvertrag für Software auf Mietbasis über Internet stellt verschärfte Bedingungen an die Leistungserfüllung eines ASPs. Während sich ein Softwarehersteller in einem Wartungsvertrag für herkömmliche Software üblicherweise lediglich für Updates verpflichtet und demselben eine Haftung für fehlerhafte Programme nur im grob fahrlässigen Fall zugesprochen wird, besteht für den ASP während der gesamten Laufzeit aller seiner Kundenverträge eine Verpflichtung zur Leistungserfüllung. Das Internet stellt in diesem Zusammenhang eine unbeeinflussbare, permanente und riskante Größe dar. Dies fällt um so mehr ins Gewicht, wenn in den Service Level Agreements (SLAs) eine Mindestverfügbarkeit garantiert wird.

Große Sorgfalt und Verantwortung obliegen einem ASP außerdem beim Umgang mit vertraulichen Kundendaten. Wie bereits angesprochen, ist Diskretion oberstes Gebot und sollte so auch entsprechend in den SLAs definiert werden.
Um das Kundenvertrauen zu halten, müssen ASPs bestrebt sein, ihre SLAs einzuhalten.

- **Vertrauen und Kontrolle**

Bevor ein ASP Partnerschaften eingeht, sollte er sich seine zukünftigen Partner und dessen Produkte genau ansehen. Schließlich muss der ASP dem Kunden auch Leistungen garantieren, die nicht in seinem eigenen Kompetenzbereich liegen. Denn für den Kunden steht die Anwendung im Vordergrund, die Aufwendungen des ASPs und seiner Partner bleiben ihm verborgen.

Qualitätssicherungsmaßnahmen, wie zum Beispiel Qualitätskontrollen bei den von den Partnern angebotenen oder angewendeten Produkten, können helfen, die Erwartungen des ASPs und seiner Kunden nicht zu enttäuschen.

Verlässliche Partner zu finden, stellt daher eine Herausforderung für ASPs dar.

- **Softwarepiraterie und Hacker**

ASPs müssen sich bewusst sein, dass diese neue Form der Softwarenutzung völlig andere Daten- und Applikationensicherungsmaßnahmen erfordert. Ein Hauptproblem für die Softwareindustrie stellt die Softwarepiraterie dar. Während beim Vertrieb von Software auf Datenträgern als Maßnahmen gegen Raubkopierer Kopierschutzverfahren, Hardware-Dongles und Lizenzierungsverfahren entwickelt und wenig erfolgreich angewendet werden, liegt das Hauptaugenmerk bei ASP bei den Autorisierungsmaßnahmen und

Zugangsbeschränkungen. Da lauffähige Softwarekopien durch die Mehrbenutzerfähigkeit implizit in jeder ASP-Software vorhanden sind, wird ein Raubkopierer versuchen, sich dazu Zugang zu verschaffen.

Unabhängig von der Softwarepiraterie gibt es aber noch eine andere gravierendere Gefahr, nämlich die der Hackerangriffe. Bei herkömmlicher Soft- und Hardware, die ausschließlich offline arbeitet (d.h. ohne Internetanschluss und -zugang), sind Hackerangriffe wesentlich unwahrscheinlicher. Bei ASP hingegen ist die Gefahr von Hackerangriffen durch den permanenten Internetzugang evident. Erschwerend kommt hinzu, dass Hacker im Unterschied zu Raubkopierern zusätzlich zum illegalen Applikationszugang potentiell in der Lage sind, auch auf fremde Applikationsdaten (Kundendaten) Zugriff zu nehmen.

Bei einer In-House-Lösung, die nicht am Internet hängt, sind Datenangriffe hingegen nur durch Mitarbeiter oder andere Personen, die direkten physischen Kontakt zu den Daten und Applikationen haben, denkbar. Die Gefahr von Datenmissbrauch durch Mitarbeiter wird in ihrer Art nicht verändert, sondern von den Mitarbeitern eines Unternehmens zu den Mitarbeitern eines ASPs verlagert.

Im Kapitel "Sicherheit" werden die eben genannten Aspekte einer praktischen Analyse unterzogen.

1.3.3. Technologiesicht

In diesem Unterkapitel soll kurz erklärt werden, was ASP aus technologischer Sicht ist und wie es prinzipiell funktioniert.

ASP zwingt nicht zu konkreten technischen Realisierungen. Jedoch ermöglichten erst die technischen Entwicklungen der letzten Jahre das Aufkommen der Branche. Letztendlich bestimmen die einzelnen Applikationsarten und Organisationsformen von ASP die einzusetzenden Techniken. In der gegenwärtigen Phase der Nicht-Standardisierung stellt sich für ASPs als zentrale Frage die der einzusetzenden Soft- und Hardwaretechnologien und Programmiersprachen. Jedem, der sich am ASP Geschäft beteiligt, sollten die technischen Anforderungen an Soft- und Hardware bewusst sein. Besonders für Soft- und Hardwareentwickler liegt das primäre Augenmerk auf der Technologiesicht.

Unabhängig von Umfang und Komplexität der Anwendungen gibt es jedoch, der Merkmalsdefinition von ASP von Kapitel 1.1. Folge leistend, eine technische Minimalfunktionalität, die alle ASP-Lösungen gemeinsam haben müssen. Es sind hierbei die Punkte der zentralen Speicherung, Verwaltung und Echtzeitbereitstellung von Applikationen und Daten und des webbasierten Zugriffs von Bedeutung.

Das technische Prinzip von ASP ist dabei immer gleich: Der Client übergibt Anforderungen (Tastatureingaben, Mausklicks und sonstige Daten) an serverseitige, mehrbenutzerfähige Web Services oder Applikationen und verwertet Ergebnisse (siehe auch [SCN Education 2000] S.16). Dabei spielen Protokolle zum Datenaustausch eine große Rolle. Die Verwertung der Daten durch den Client kann entweder durch Visualisierung erfolgen (hierbei werden vom Server Browserprogrammcode wie HTML, JavaScript oder Java oder "Zeichenregeln" zur Visualisierung mitgeschickt) oder durch Konsumation durch clientseitige Programme. ASP Dienste, bei denen die Präsentationslogik optional ist und die sich auch von anderen Client-Programmen als einem Internetbrowser konsumieren lassen, werden z.B. von Microsoft auch mit "Web Services" [Schaub 2000] bezeichnet, sind jedoch aus technischer Sicht ebenfalls vollwertige ASP Lösungen. ASP Softwaretechniken, die eben besprochene Grundfunktionen ermöglichen, werden mit "ASP enabling platforms" und "ASP development platforms" (in dieser Diplomarbeit zu "ASP Basistechnologien" zusammengefasst) bezeichnet.

In der Praxis zeigt sich jedoch das gesamte Ausmaß der technischen Komplexität von ASP, besonders bei kommerziellen Anwendungen. Technische Infrastruktur, die zusätzlich zum ASP-Betrieb benötigt wird, wird mit "ASP supporting technologies" bezeichnet.

Der Hauptteil dieser Diplomarbeit behandelt im Kapitel "Organisation und Technik" umfassend solche für ASP relevanten Techniken.

1.4. ASP – Player

Der Markt für Anbieter im ASP-Geschäft ist sehr groß: Firmen aus vielen Bereichen (nicht nur IT) sehen Chancen, ihr Know-how, ihre Leistungen und ihre Produkte zu verkaufen. Prinzipiell gibt es für diese Firmen die Möglichkeit, einen ASP zu beliefern oder selbst ein ASP zu werden, sei es als Haupttätigkeit oder als Nebentätigkeit. Ein ASP sein heißt, wie bereits erwähnt, Alleinverantwortlichkeit für ein ASP Produkt gegenüber dem Endkunden, also dem Anwender, zu übernehmen. Wie und mit wem ein solches ASP Endprodukt zustande gebracht wird, ist dem ASP freigestellt und daher ein großes Entscheidungsfeld.

ASPs, Partnerfirmen und Lieferanten unterscheiden sich durch ihre Kompetenzen, Geschäftsmodelle und ihr Angebot an Produkten. Ein Endprodukt für Kunden entsteht durch Zusammenführung dieser Services zu einer sog. "ASP Lösung". Dieses Kapitel soll zeigen, welche Anbieter es gibt, welche Rolle sie im ASP Geschäftsbereich spielen und in welche Kategorien sie sich, abhängig von den angebotenen Leistungen, einteilen lassen.

1.4.1. Kernkompetenzen, Geschäftsmodelle und Produktkategorien

Ein geeigneter, allgemeiner Weg, um deutlich zu machen, welche Anbieter sich im ASP-Bereich befinden und einfinden werden, ist die Angabe von Kernkompetenzen.

IDC [Gillan et al. 1999] teilt sie in drei Bereiche ein: Dienstleistungen (Services), Netzwerke (Network) und Anwendungen (Application):

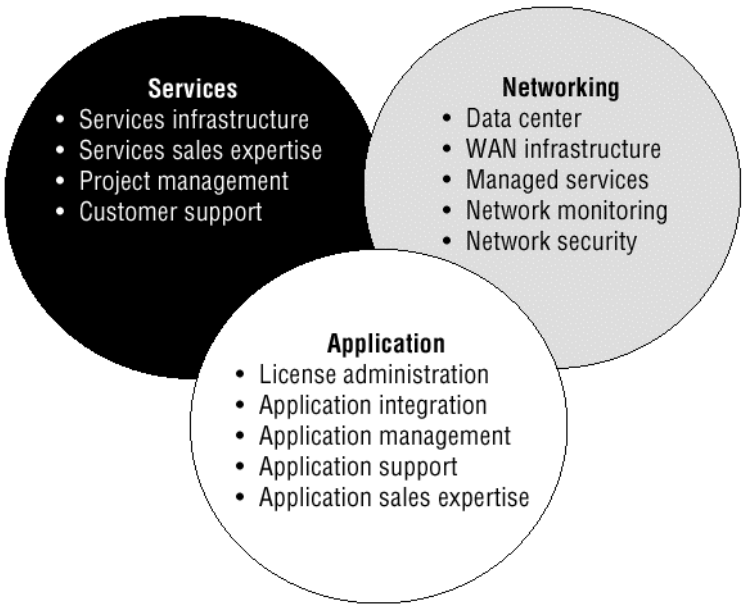

Abbildung 1-3 : ASP – Die benötigten Fähigkeiten [Gillan et al. 1999]

Je nach den Anforderungen und der Qualität einer ASP-Lösung werden diese Kompetenzen und Leistungen mehr oder weniger stark benötigt, manche können im einzelnen optional oder sogar nicht erforderlich sein.

Potentiell besteht aber für jedes Unternehmen, das zumindest eine der in der Grafik angeführten Kompetenzen vorweisen kann, die Möglichkeit, am ASP-Geschäft teilzuhaben, aber die Liste der benötigten Fähigkeiten lässt sich aufgrund der undefinierten Marktsituation noch nicht vollständig angeben.

Auf diese Teilhaber des ASP-Geschäfts wird in der Folge näher eingegangen.

1.4.2. ASP - Player Schichtenmodell

Farleit Limited [Farleit 2000] und Lixin Tao [Tao 2000] teilen die Struktur einer ASP-Lösung und die daran beteiligten ASP-Player in mehrere aufeinander aufbauende Schichten auf.

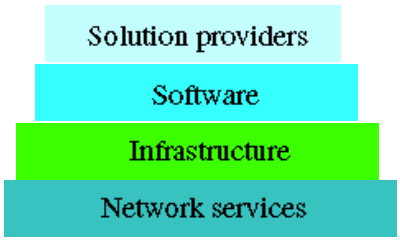

Abbildung 1-4 : ASP Schichtenmodell [Tao 2000]

Diesen Ebenen sind direkt spezialisierte Provider zuordenbar, es ist aber auch denkbar, dass ein Provider mehrere oder der ASP selbst (der "Solution provider") mehrere oder alle Schichten abdeckt. Jeder Provider einer Schicht macht sich die Leistungen und Technologien der weiter unten liegenderen Schichten zunutze. Begonnen mit der untersten Ebene sind dies:

- **Network services**

  Auf der Netzwerkschicht sitzen die Anbieter von Netzwerkdiensten wie Kommunikation, Server Ressourcen und "Value-added" IP-Diensten. Zur Kommunikation zählen Netzwerkressourcen auf physikalischer Ebene (Kabel, Repeater, Switches, Bridges, Router etc.) und das damit verbundene Netzmanagement (zur Regelung von Kommunikation, Performance, Ausfallssicherheit und Security). Zu den Server Ressourcen zählen typischerweise Webspace, unterbrechungsfreie Stromversorgungen (USVs), Einrichtungen für physikalische Sicherheit (z.B. Chipkartenlesegeräte) und technische Netzwartungsdienste. "Value-added" IP-Dienste inkludieren "Virtual Private Networking" (VPN) , Netzwerk-Caching, Firewalls und Directory Services.

  Anbieter von Network Services werden mit **Network Service Provider (NSP)** bezeichnet.

- **Infrastruktur**

  In dieser Schicht operieren die Anbieter von ergänzenden Soft- und Hardwarekomponenten und Dienstleistungen, die weitere Voraussetzungen für den ASP Betrieb gewährleisten. Die Hersteller bieten diese Produkte dem ASP zum Verkauf oder zur Vermietung an. Viele Anbieter betreiben die Produkte in ihrem eigenen *Data Center*.

  Hard- und Softwareprovider bieten in diesem Sektor z.B. Storage-Lösungen an, auf die der ASP via Internet die Kunden- und Protokolldaten sichern kann.

Sie bieten weiters Dienstleistungen wie technischen Support für ASPs oder ASP-Kunden, Ausbildungsprogramme für das ASP-Personal, übernehmen das Call Center und helfen mit Finanzierungsprogrammen.

Es gibt aber auch Infrastruktur-Anbieter, die den gesamten ASP-Betrieb in ihren Data Centers übernehmen. Diese **ASP Enabler**- oder **Application Infrastructure Provider (AIP)**-Rolle inkludiert die Koordination der Netzwerkdienste und des Systemmanagements, die Bereitstellung, den Betrieb und das Management der Systemhardware und -software, das Management der ASP-Kunden Accounts, der Abrechnung und die Kundenunterstützung. Wichtige weitere Aufgaben als AIP sind das Applicationmanagement, das Service Level Monitoring, das Schaffen der Infrastruktur für den Help-Desk und die nahtlose Übermittlung von Statusinformationen über den ASP-Betrieb an die Partner innerhalb des ASP-Schichtenmodells. Um das Serviceangebot zu komplettieren bieten AIPs Schritt-für-Schritt Programme zur Umstellung herkömmlicher Software auf die ASP-Umgebung. Sie erlauben den Kunden (die in diesem Fall die ASPs darstellen) außerdem ihre Lösungen zuerst in einer Test-Umgebung laufen zu lassen, bevor die Applikationsdienste tatsächlich von den Endkunden genutzt werden.

- **Software**

Software Provider ermöglichen erst die Erfüllung der Hauptaufgabe von ASP, nämlich das Bereitstellen der Applikationsfunktionalität an den Endkunden. Softwarehersteller liefern einerseits spezielle "Applikationserver" und ASP-Entwicklungsplattformen und -tools für Entwickler (siehe Kapitel 1.3.3.) und andererseits die eigentlichen Applikationen. Die Applikationen können herkömmliche Programme sein, die für den ASP-Betrieb angepasst wurden, oder es können speziell für diesen Zweck entwickelte Programme sein. Obwohl es bereits eine Reihe von Applikationsservern gibt, haben nur wenige das komplette Set an Services, das für einen ausgereiften ASP-Betrieb erforderlich ist, implementiert. Dazu zählen die softwaretechnischen Aufgaben der Servicebereitstellung, des Subscribermanagements, der Kundenunterstützung, des Service Level Managements, der Security und der Kundenverrechnung. Verschiedene Softwarehersteller bieten aber spezielle Produkte für die jeweiligen Aufgabenbereiche. Herausforderungen sind die nahtlose Zusammenarbeit mit den Applikationsservern, den eigentlichen Applikationen und allen anderen Komponenten des ASP-Schichtenmodells (**Integration**).

- **Solution Provider**

Solution Provider ist eine andere Bezeichnung für **Application Service Provider**. Diese Rolle des ASP ist nicht einfach abzugrenzen. In der Praxis gibt es Unternehmen, auf die Definition 1.1. zutrifft, die sich jedoch nicht als ASP bezeichnen, und es gibt umgekehrt Firmen, die sich ASP nennen, aber auf die die Definition nicht voll zutrifft. Eine Gemeinsamkeit in der Aufgabenverteilung von ASPs lässt sich jedoch finden: Der ASP ist in jedem Fall derjenige, der die Beziehung zum Endkunden betreut. Die Applikationsbenutzungsrechte werden in den meisten Fällen vertraglich geregelt (Mietvertrag), bei kostenlosen Angeboten gibt es hingegen kein Vertragsverhältnis und die Endkunden haben in diesem Fall keinen Rechtsanspruch auf Leistungen. Seriöserweise sollten sich ASPs nur als solche bezeichnen, wenn sie ihre Leistungen auch vertraglich zusichern können, das bloße Einräumen einer Nutzungsmöglichkeit einer Applikation über Internet macht noch keinen ASP aus. Professionelle ASPs bieten zudem Service Level Agreements (SLAs) an, das sind vertraglich zugesicherte Mehrleistungen.

Das Zusammenführen der Leistungen der einzelnen Anbieter, welche sich in den eben zitierten Schichten bewegen, und das Schnüren von Applikationslösungen und die Vertragsgestaltung für Endkunden sind also die Hauptaufgaben des ASPs.

Andere Aufgaben müssen nicht notwendigerweise vom ASP selbst erledigt werden und können von einem oder mehreren AIPs übernommen werden (Outsourcing).

1.4.3. ASP - Player Mengenmodell (Rollen)

Ein anderes Modell, das die unterschiedlichen Rollen, die die im letzten Kapitel beschriebenen Unternehmen in Bezug auf das ASP einnehmen, aufzeigt, ist in Abbildung 1-5 [Gillan et al. 1999] zu sehen:

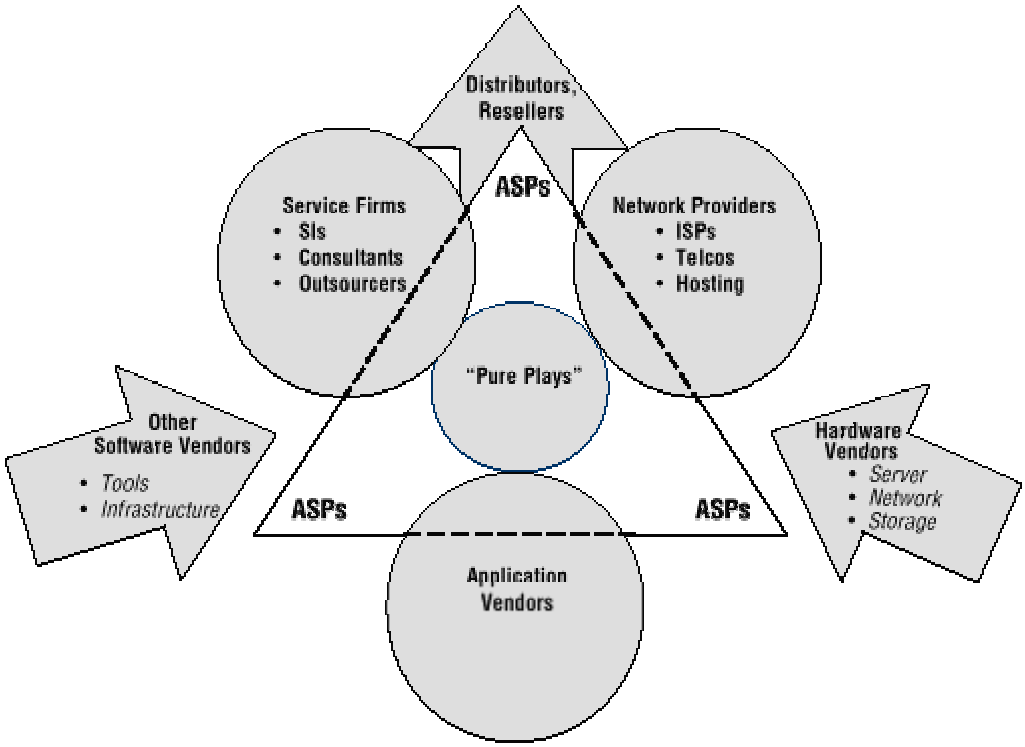

Abbildung 1-5 : Rollen von IT-Unternehmen im ASP-Geschäft [Gillan et al. 1999]

Dabei werden drei Rollen unterschieden:

Vollständig innerhalb des Dreieckes befinden sich jene ASPs, die ASP als ihre Hauptaufgabe sehen. Dazu gehören die "Pure Plays" (Full Service Provider), oftmals "Start-Ups“, die gleich als ASPs auftreten. In den großen Kreisen sind die Softwarehersteller ("Application Vendors"), Hoster/Netzwerkanbieter ("Network Providers") und Dienstleister ("Service Firms") abgebildet. Stammt ein Unternehmen aus einem dieser Bereiche und bietet es nun Software zur Miete an, so befindet sich dieses Unternehmen in dem Teil des Kreises, der sich innerhalb des Dreieckes befindet und kann sich somit als "ASP" bezeichnen. Die übrigen werden als Partner von ASPs betrachtet und befinden sich in dem Teil des Kreises, der außerhalb des Dreieckes liegt.

Lieferanten von Hardware und Softwaretools für ASPs sind in großen Pfeilen dargestellt. Von ihnen wird am wenigsten erwartet, dass sie selbst als ASP auftreten werden, da sie sich dann zu weit von ihren Kernkompetenzen entfernen würden.

Als nächstes werden die ASPs nach den angebotenen Lösungen und Dienstleistungen kategorisiert.

1.4.4. Arten von Application Service Providern

Eine Einteilung von ASPs in bestimmte Kategorien ist schwierig, da viele Kriterien zur Unterscheidung von ASPs existieren. So lassen sich ASPs z.B. nach der Art der angebotenen Software unterscheiden. Software lässt sich z.B. nach horizontaler und vertikaler Ausrichtung unterscheiden (siehe Kapitel 1.3.2 , "Branchenwissen"), aber auch z.B. nach dem Grad der Komplexität. Weiters lassen sich ASPs nach der Art ihrer Geschäftsmodelle oder nach dem Geschäftserfolg unterscheiden. Auch eine Einteilung nach der Zielgruppe ist möglich: Das Angebot eines ASPs kann auf einzelne oder mehrere Unternehmen, auf Privatkunden oder auf Regierungen zugeschnitten sein.

Jedes Angebot zur Softwaremiete über Internet wird außerdem maßgeblich von Art und Umfang der bereitgestellten Dienstleistungen beeinflusst (wird in den SLAs definiert: Art der Abrechnung, Existenz und Art eines Call Centers, Wartungsintervalle, Entschädigungen für Ausfälle etc.).

Das Modell von IDC ([Gillan et al. 1999], zu sehen in Abbildung 1-6) versucht, (fast) alle eben aufgestellten Unterscheidungskriterien zusammenzufassen und in Relation zu stellen:

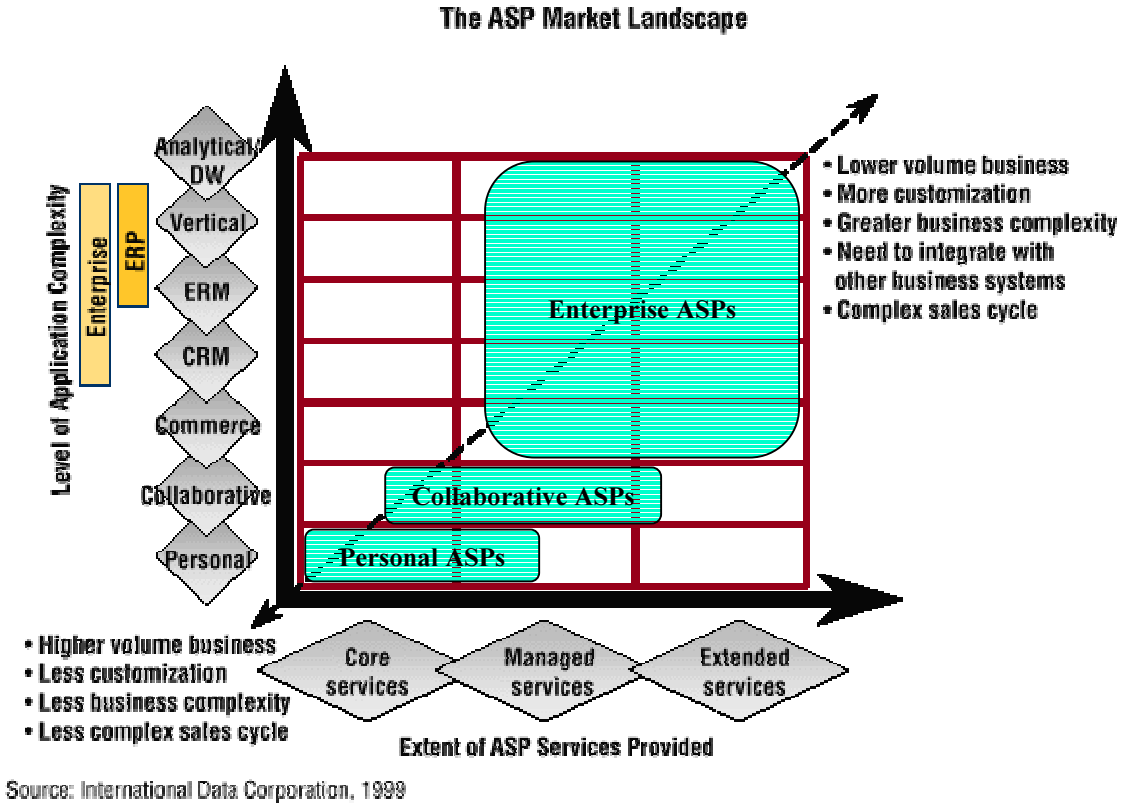

Abbildung 1-6 : Application Service Provider Marktlandschaft [Gillan et al. 1999]

Dieses Spielfeld wird zum einen durch den Grad der Komplexität der Anwendung und zum anderen durch das Ausmaß der angebotenen Dienstleistungen aufgespannt. An der Ordinate sind die Anwendungsarten aufgetragen. Die Abszisse unterteilt die Application Service Provider nach dem Umfang ihrer Dienstleistungen. Da sich der Markt noch entwickelt, ist diese Einteilung als Prognose zu sehen. Danach werden die Angebote in eine von den folgenden drei Kategorien fallen [Gillan et al. 1999][Berheide 2000] :

- **Core Services**

  Diese Dienstleistungen umfassen lediglich grundlegende Leistungen, um die Softwareumgebung zu pflegen und somit ein Mindestmaß an Kundenzufriedenheit zu generieren. Zu nennen wären regelmäßige Updates und Upgrades der Applikationen, 24 Stunden-7 Tage der Woche-365 Tage im Jahr Aufsicht und Kontrolle der Software, des Netzwerkes und der Server, sowie minimale Kundenbetreuung.

- **Managed Services**

  Diese beinhalten die Core Services und darüber hinaus Dienstleistungen und Garantien für die Unterstützung, Sicherheit, Softwareleistungen und Datenredundanz. Hinzu kommen Service Level Agreements (SLAs) über die Softwareperformance und Datensicherheit, reserviertes technisches Personal und tägliches Backup von Daten.

- **Extended Services**

  Dies sind Managed Services und darüber hinausgehende Leistungen. Zu nennen wären individuelle Konfiguration und Unterstützungen in den Bereichen Strategie und Planung, sowie Training und Ausbildung.

ASP Angebote positionieren sich größtenteils in den grauen Quadranten innerhalb der Matrix und charakterisieren die ASPs als "Enterprise ASP", "Collaborative ASP" oder als "Personal ASP":

- **Enterprise ASP**

  Die Enterprise ASPs sind Highend-Anbieter. Sie befinden sich im oberen rechten Quadranten des Modells. Sie sind spezialisierte Anbieter mit hohem Applikationen- und Branchen Know-how, und haben Erfahrungen auf dem Gebiet der Softwareintegration. Die Enterprise ASPs bieten analytische und branchenspezifische Software, sowie komplexe ERP- und CRM-Systeme an.

  Unternehmen, die solche ASP-basierte Software herstellen, sind bspw. SAP, Siebel, Oracle, J.D.Edwards und Peoplesoft. International bekannte ASPs, die in diesem Bereich zu finden sind, sind z.B. Corio, USInternetworking, Interpath, Agilera oder NaviSite Inc.

- **Collaborative ASP**

  Collaborative ASPs sind im mittleren Segment positioniert. Diese Unternehmen bieten weniger komplexe Software wie z.B. Groupware, Conferencing oder Unified Messaging an. Auch Projektmanagement-Lösungen und Sales Automation (Online-Shops) gehören in das Portfolio dieser ASPs. Für diese Software leisten sie Managed Services. Unternehmen, die in diesem Bereich tätig sind, müssen sich mehr auf die Verfügbarkeit der Anwendungen konzentrieren, da diese Software nicht die Integrationsleistung benötigt, wie vergleichsweise eine Software zur Automatisierung von Geschäftsabläufen.

  Hersteller solcher Software sind Microsoft (MS Exchange), Lotus (Notes) sowie Online-Shop-Software von Intershop, Arriba, Broadvision oder Commerce One.

  Anbieter (ASPs) dieser Software sind bspw. Mi8, Interpath und Critical Path.

- **Personal ASP**

  Personal ASPs sind im unteren linken Bereich positioniert. Sie bieten einfache, sofort nutzbare Lösungen (Office-Software, Termin- und Reiseplanung, E-Mail, Adressenverwaltung, DTP (Desktop Publishing); siehe auch Kapitel 1.5., "Personal Applikationen") an. Sie haben besondere Fähigkeiten, die hohe Kundenzahl und das damit verbundene hohe Datenaufkommen zu betreuen und managen die Server und Netzwerke, auf denen diese Software laufen.

  Wichtige Softwarehersteller in diesem Bereich sind z.B. Microsoft, Sun und Lotus.

  Zu den Personal ASPs zählen beispielsweise personable.com, C Me Run, FutureLink, Interliant, USInternetworking und Verio.

1.5. Applikationen und Adressaten

Als unmittelbare Frage stellt sich, welche Arten von Software sich eigentlich über einen ASP nutzen lassen. In diesem Zusammenhang wären nicht nur technisch-organisatorische sondern auch wirtschaftliche Überlegungen zu nennen. Im Rahmen der Behandlung der technisch-organisatorischen Konzepte von ASP im Kapitel 2 und den wirtschaftlichen Überlegungen in Kapitel 1.6. wird dem Rechnung getragen, es würde jedoch den Rahmen dieser Arbeit sprengen, zu jeder möglichen Softwareart eine ideal-technisch-organisatorische und wirtschaftliche Form von ASP herauszuarbeiten, zumal der Markt auch noch nicht definiert ist.

Einer Prognose von IDC zufolge (wird zitiert in [Berheide 2000]) werden die ASP-Angebote in folgenden Kategorien zu finden sein, begonnen mit der komplexesten:

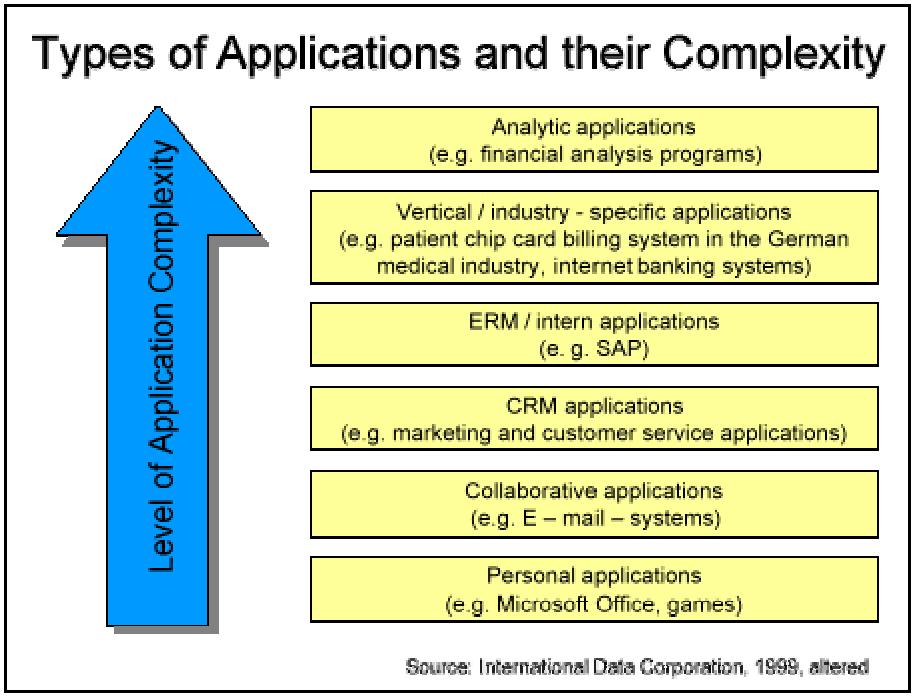

Abbildung 1-7 : Applikationsarten und deren Komplexität [Berheide 2000]

- Analytische Anwendungen. Diese umfassen alle Anwendungen, die Unternehmensprobleme analysieren können (z.B. Finanzanalysen, Analyse der Webseite, Risikoanalysen, o.ä.).
- Vertikale Applikationen. Diese beinhalten sämtliche branchenspezifische Applikationen, wie z.B. Patientenabrechnungen im Krankenhausgewerbe oder Prozesssoftware in einer Versicherungsanstalt.
- ERP-Software. Diese übernehmen Aufgaben in den Bereichen Rechnungswesen, Personal, Facility Management und/oder Lagerhaltung.
- CRM-Software. Diese Applikationen umfassen Funktionen wie z.B. Kundenservice, Marketingapplikationen und Sales Force Automation.
- Kollaborative Anwendungen. Das sind Anwendungen wie E-Mail, Groupware und Konferenzsoftware.
- Personal Applikationen. Hier zu nennen sind Programme wie Microsoft Office und andere Software für Konsumenten (Spiele, Kalender, etc.).

Ähnlich schwierig zu beantworten ist die Frage nach der Zielgruppe: Zu Beginn der Outsourcing- und ASP-Ära wurden die KMUs als Zielgruppe angegeben. Motive hierfür wären:

- Meistens fehlt eine eigene IT-Abteilung, oder es fehlt an ausgebildeten IT-Spezialisten.
- KMUs können die Chance ergreifen, sich teure Software ohne hohe Vorabinvestitionen leisten zu können und somit mit größeren Unternehmen zu konkurrieren.
- Unter den KMUs finden sich "Start-Ups", für die das Internet und vor allem dessen Einbeziehen in die eigenen Geschäftsprozesse nichts Außergewöhnliches ist [Igler 1999].

Und doch ist der Markt viel größer (Evans Marketing Services (EMS), wird zitiert in [Igler 1999]): Immerhin denken in den USA gut ein Drittel der IT-Manager in großen Unternehmen ebenfalls nach, in Zukunft intensiv mit ASPs zusammenzuarbeiten. Aufgebrochen nach Branchen ergibt sich bei den Versicherungsunternehmen ein noch deutlicheres Votum zugunsten der neuen Möglichkeiten: Hier liegt die Bereitschaft zur Zusammenarbeit gar bei 45 Prozent. Im Finanzbereich tendieren 34,1 Prozent zu ASP, im Gesundheitswesen 32 Prozent. Im Rahmen der Umfrage wurden 400 Manager von Unternehmen mit mehr als 2000 Mitarbeitern befragt.

Sobald ASP auch für den Massenmarkt verfügbar wird (z.B. kostenlose Angebote, durch Werbung finanziert), wird ASP auch für Privatanwender interessant werden. Als Anbieter werden hierfür auch die Regierungen und Institutionen (z.B. Banken) hinzukommen [Kennedy 2000].

Aus diesem kurzen Überblick ist ersichtlich, dass einerseits die Art und Komplexität der Anwendung stark mit dem Umfang der ASP-Services korreliert und andererseits der Markt entsprechend groß ist, um auf unterschiedliche Zielgruppen speziell zugeschnittene ASP-Lösungen zu positionieren.

Einen Einblick in das derzeitige Angebot an Applikationen bietet die Internetseite http://www.runapps.com. Es handelt sich dabei um eine ausgezeichnete Suchmaschine (genaugenommen um ein sog. "Portal") für ASP-Applikationen. Neben der Möglichkeit, Preise zu vergleichen, kann man kostenlose Applikationen sofort testen.

1.6. Wirtschaftliche Aspekte

Kaum einem anderen Bereich in der IT-Service Branche wird eine solch hohe wirtschaftliche Bedeutung wie dem Application Service Providing zugemessen.

Zum einen sind die Gründe hierfür die breite Palette an Anbietern, die im ASP Bereich investieren, zum anderen betriebswirtschaftliche Überlegungen von ASP Kunden. Diese Relation aus Angebot und Nachfrage bestimmt den Markt.

Einige Betrachtungen der Marktstruktur, die den wirtschaftlichen Wert von ASP offen legen, wie Arten der Anbieter, Arten der Anwendungen, Chancen und Risiken für ASP-Player sowie Vor- und Nachteile für ASP-Kunden wurden bereits vorgenommen.

Diese Teilbereiche werden nun einerseits durch die Wertschöpfungskette von ASP, die aktuelle Marktsituation und die Marktentwicklung im ASP Bereich und andererseits durch betriebswirtschaftliche Aspekte von ASP Kunden, in erster Linie Kostenvorteile, ergänzt.

Allerdings wurde auf eine Darstellung von optimalen Unternehmensstrategien für ASP-Player und ASP-Kunden verzichtet, zumal die kritischen Erfolgsfaktoren erst bei der Konsolidierung des Marktes sichtbar sein werden. Eine Reihe von strategischen Empfehlungen sind in [Arthur D Little 2000], [Cherry Tree 1999], [c-quential 2000], [Farleit 2000], [Gillan et al. 1999] und [Summit 1999a] zu finden.

1.6.1. Wertschöpfungskette

Um zu zeigen, wie mit dem ASP Modell Geschäfte gemacht werden, ist eine Darstellung der ASP-Wertschöpfungskette dienlich [Compaq 2001]:

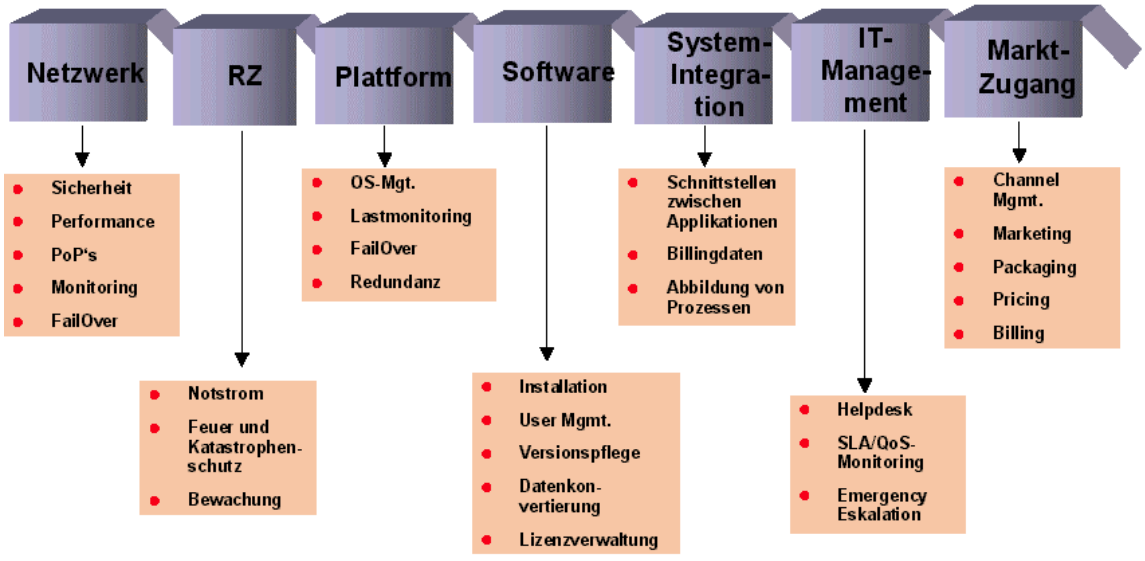

Abbildung 1-8 : Wertschöpfungskette im ASP-Markt [Compaq 2001]

Wie man aus der Abbildung erkennen kann, ist die Wertschöpfungskette im ASP-Markt sehr komplex und umfassend, weshalb eine Alles-aus-einer-Hand-Lösung sehr selten anzutreffen sein wird. Die beteiligten ASP Player wurden bereits in Kapitel 1.4. vorgestellt. Jeder Anbieter kann in einem oder mehreren Bereichen der Wertschöpfungskette tätig sein.

Nun sollen die Geschäftsverhältnisse der Anbieter näher erläutert werden: Der Betreiber des Rechenzentrums oder Datacenters ("RZ") kauft oder mietet und installiert Komponenten eines Netzwerkanbieters und betreibt danach das Netzwerk selbst oder überträgt diese Aufgabe dem Network Service Provider. Er installiert und betreibt die Soft- und Hardwarekoponenten, die für

die ASP-Infrastruktur erforderlich sind oder überträgt diese Aufgabe einem Dienstleister ("Plattform"). Dazu bezieht das Rechenzentrum ASP Basistechnologien und optional auch ASP Applikationen vom ISV ("Software"). Der ASP ("Marktzugang") zieht Systemintegratoren ("Systemintegration") zu Rate, um die Software an die jeweiligen technischen und geschäftlichen Bedürfnisse anzupassen, oder übernimmt selbst diese Aufgabe. Die ASP Applikationen implementiert der ASP entweder selbst, bezieht sie vom ISV oder mietet sie vom Rechenzentrum. Die Prozesse des Operation Managements (siehe Kapitel 2.6.4.) und der Kundenbetreuung kann der ASP ebenfalls an einen anderen Service Provider auslagern. Am Ende der ASP-Wertschöpfungskette (rechts neben "Marktzugang") steht der Kunde. Dieser mietet das sich daraus ergebende Endprodukt, die ASP Lösung, vom ASP.

Aus der ASP-Wertschöpfungskette ist das Marktpotential des ASP Modells gut ersichtlich.

1.6.2. Marktsituation und Marktentwicklung

Der ASP-Markt ist von nahezu ständigem Wachstum geprägt: Die Schätzungen des weltweiten ASP-Marktvolumens reichen von 11,3 Mrd. $ in 2003 (Forrester Research) bis 22,5 Mrd. $ in 2003 (Gartner). Die Marktforschungsinstitute IDC, Gartner und Forrester geben etwas unterschiedliche Prognosen [Compaq 2001]:

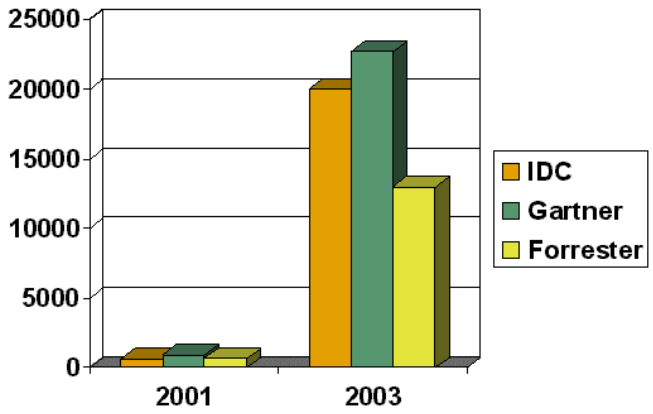

Abbildung 1-9 : Weltweiter ASP-Umsatz (in Millionen US-$) [Compaq 2001]

Die Entwicklung des europäischen ASP-Marktes hinkt der Entwicklung in den USA ca. 2 Jahre hinterher, aber auch hier sind hohe Wachstumsraten zu erwarten. Für Deutschland, Österreich und die Schweiz prognostiziert FORIT folgende Aufteilung nach Anwendungsarten [Bauer 2001]:

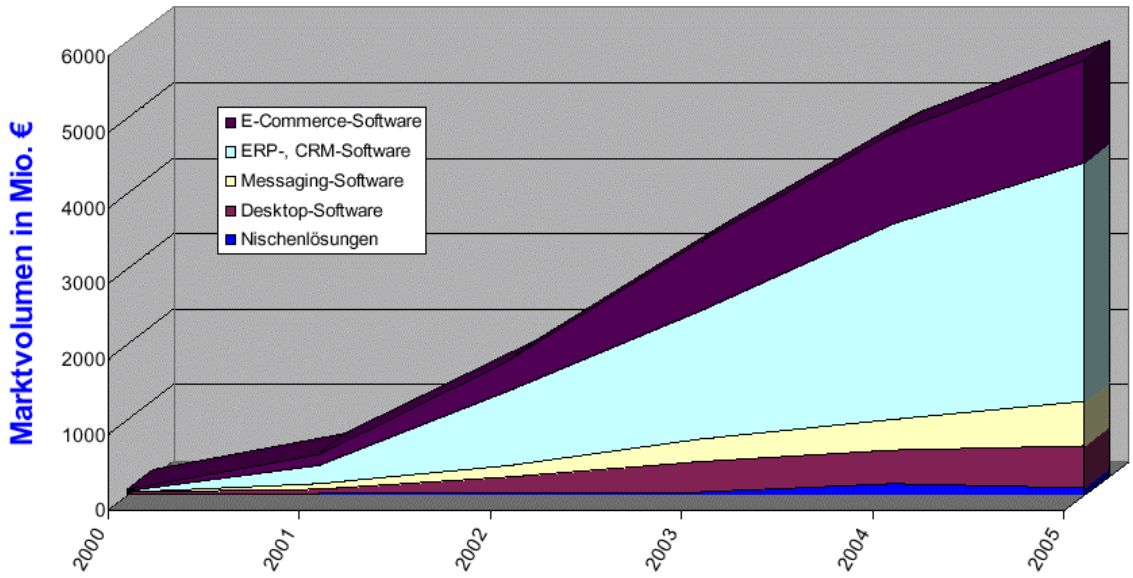

Abbildung 1-10 : Prognosen für das ASP-Marktvolumen in D, A und CH [Bauer 2001]

Etwas detaillierter zum weltweiten Wettbewerb im ASP Markt äußert sich die META Group (zitiert in [Compaq 2001]):

Die Jahre **2000**, **2001** und **Anfang 2002**, also die jetzige Marktsituation, sind zu der Phase der **Marktentstehung** hinzuzurechnen. Es ist eine Marktphase, die noch keine geordneten Strukturen trägt, aber schon von Konsolidierungen geprägt ist. Partnerschaften werden geknüpft, um den Kunden möglichst alle Teile der Wertschöpfungskette aus einer Hand bieten zu können. Mögliche Kunden sind noch sehr zurückhaltend, da sie noch kein Vertrauen in ASP-Anbieter haben. Es existieren viele Start-Ups, die sich als ASP-Anbieter versuchen. Die ASP-Anbieteranzahl (derzeit über 500 ASPs weltweit) übersteigt das Maß, das der Markt verträgt. Differenzieren können sich ASP-Anbieter hier am ehesten durch eine starke Servicequalität. Markennamen werden hier definiert und im Markt gesetzt. Markteintrittsbarrieren sind in dieser Marktphase gering.

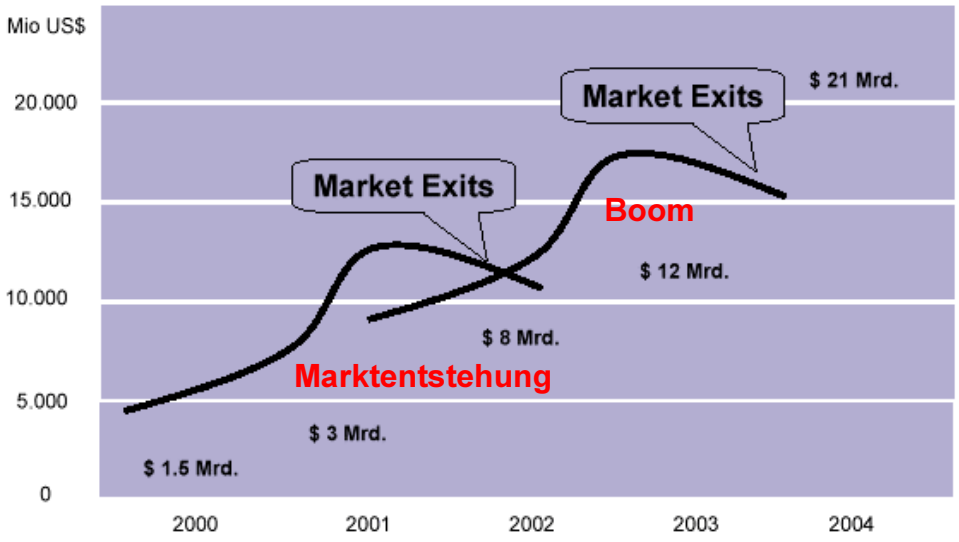

Abbildung 1-11 : Entwicklung der weltweiten Umsätze [Compaq 2001]

2002 und **2003** wird es einen regelrechten **Boom** geben: Nachdem zwischen 50% und 60% aller derzeitigen ASPs aufgrund mangelnder Geschäftsmodelle vom Markt verschwunden sein

werden, werden sich die übriggebliebenen ASPs durch starke Kompetenzen und hohe Umsätze auszeichnen. Die Applikationsangebote werden sich durch höhere Komplexität und Service Levels auszeichnen.

Ab **2004** wird die **Reifephase** eintreten: Etablierte Geschäftsmodelle setzen sich durch, "Kinderkrankheiten" werden beseitigt. ERP und CRM Systeme werden auf Grund der Komplexität erst dann massiv wachsen.

Für die META Group ist das am stärksten wachsende Segment das E-Business ASP. Hierfür ausschlaggebend sind die für diesen Bereich hohen Anforderungen an speziellen Know How und die sehr kurze Einführungsphase.

1.6.3. Betriebswirtschaftliche Aspekte

Die Zielgruppe von ASP stellen in erster Linie Unternehmungen dar, respektive KMUs. Die betriebswirtschaftlichen Auswirkungen für solche ASP-Kunden seien nochmals zusammengefasst:

- **Niedrige Initialinvestitionen**

  Die Einrichtungsgebühr für eine ASP Applikation beträgt meist nur einen Bruchteil der Anschaffungskosten für Soft- und Hardware für eine herkömmliche Lösung.

- **Total Cost of Ownership (TCO) minimieren**

  Die ASP-Alternative erlaubt typischerweise eine jährliche Kostenreduzierung von 30% bis 50%, abhängig von der Komplexität der jeweiligen Applikation. Über 75% der derzeitigen Benutzer von ASP Lösungen bestätigen einen "Return Of Investment" innerhalb des ersten Jahres [CompTIA 2002]. Folgende Grafiken wurden [c-quential 2000] entnommen. Die Analyse basiert auf Unternehmen mit etwa 10 Endgeräten.

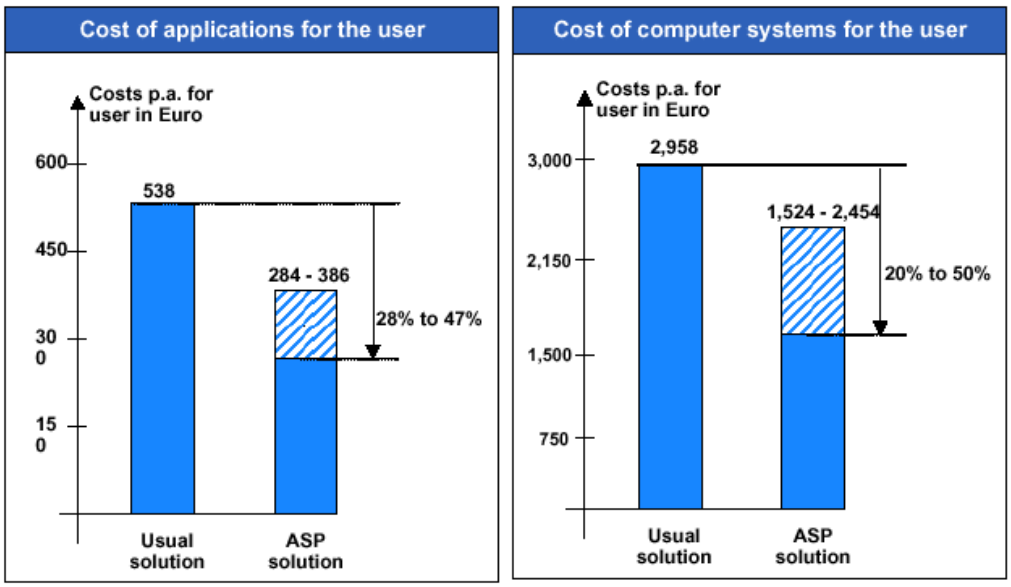

Abbildung 1-12 : Kostenreduktionen durch ASP bei Hard- und Software [c-quential 2000]

- **Kostentransparenz**

  Das ASP Konzept erlaubt einen hohen Grad an Vorhersehbarkeit der Kosten durch Wegfall von Unsicherheiten betreffend die Aufwendungen für Soft- und Hardwarewartung.

- **Effizienz des Personals verbessern**

  Der Wegfall des Applikationsmanagements erlaubt dem Personal die Konzentration auf Kernkompetenzen. Internes IT-Personal, falls vorhanden, kann sich auf die geschäftskritischen Bereiche der IT konzentrieren. In anderen Fällen kann auf IT-Personal sogar gänzlich verzichtet werden, ohne dass Mitarbeiter "nebenbei" IT-Administrierungsaufgaben erfüllen müssen. Dies erlaubt eine Reduzierung der Personalkosten.

- **Geringe Kapitalbindung und effiziente Mittelallokation**

  Mit der Sichtweise des Shareholdervalue kommt die Forderung auf, die der Unternehmung zur Verfügung gestellten Mittel auch effizient zu verwenden, respektive zu investieren. Dabei werden diese stets knappen Ressourcen denjenigen Investitionsprojekten zugeteilt, die entweder höchste Priorität haben oder den höchsten „Return on Investment“ (ROI) versprechen. Bei den laufenden Investitionen wird zunehmend auf eine geringe Kapitalbindung geachtet. Zum einen üben die Banken mit ihrer restriktiven Kreditpolitik Druck auf die Klein- und Mittelunternehmen aus. Zum andern erhöht eine große Kapitalbindung die nötige Eigen- sowie Fremdkapitalverzinsung entsprechend. Somit stellt eine „leichte Bilanz“ ein weiteres Ziel der Unternehmung dar. Es verwundert kaum, dass in den letzten Jahren Leasinggesellschaften enorme Wachstumsraten ausweisen konnten. Das sogenannte Operating-Leasing stellt eine echte Alternative zum Kauf innerhalb eines Investitionsprojektes dar. Das ASP kommt dieser Form des Leasing sehr nahe.

- **Anderes Modell der Kostenentwicklung**

  Bei einer ASP-Nutzung verläuft die Kostenkurve anders als bei herkömmlichen Lösungen. In Abbildung 1-13 sind keine Einheiten enthalten, da die Kostenentwicklung lediglich als Modell dargestellt werden soll:

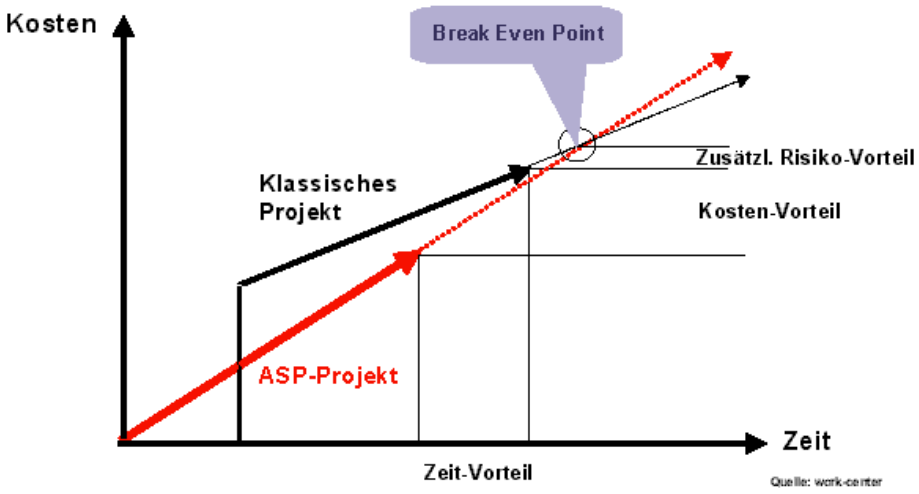

Abbildung 1-13 : Gesamtkosten einer klassischen IT-Lösung und einer ASP Lösung [Compaq 2001]

Der Anfang der Pfeile stellt jeweils den Zeitpunkt zu Beginn der Implementierphase dar. Das Ende der Pfeile stellt jeweils den Zeitpunkt des Beginns der Betriebsphase dar. ASP verkürzt die Einführungsphase und bewirkt einen Kostenvorteil beim Einsatz neuer Applikationen. Der "Break-Even-Point" stellt den Zeitpunkt dar, ab dem ein ASP-Einsatz teurer als eine In-House-Lösung ist.

- **ASP-Kosten in der Praxis**

Zum Abschluss ein Vergleich der Kosten von einer traditionellen Implementation und dem ASP-Modell am Beispiel einer Software zur Bonitätsprüfung ("Guardean"). Dabei liegen folgende Annahmen zugrunde:

- 3 Jahre Berechnungszeitraum
- Transaktions-/Nutzungsabhängige Vergebührung
- Monatliche Grundgebühr von 511 Euro
- 24000 Transaktionen pro Jahr
- Gebühr pro Transaktion beträgt 0,44 Euro
- Kosten ohne Abschreibung

		Traditionelle Implementation (Kosten in Euro)	**ASP-Modell** (Kosten in Euro)
Fixe Kosten	Hardware / Software	36.000	0
	Lizenzen	204.000	0
	Installation / Setup	51.000	7.000
Variable Kosten	Wartung	51.000	0
	Monatliche Grundgebühren	0	18.000
	Transaktions-abhängige Gebühren	0	32.000
	Summe	**345.000**	**57.000**
	Gebühr pro Bonitätsprüfung	4,79	0,79

Tabelle 2 : Kostenvergleich traditioneller Software-Vertrieb / ASP-Modell [Bauer 2001]

1.7. Rechtliche Aspekte

Während der Markterfolg des Application Service Providing immer mehr zunimmt, steht die Abhandlung rechtlicher Aspekte noch am Anfang. In Österreich existiert noch keinerlei Literatur zu diesem Thema. Es gibt insbesondere auch keinerlei Rechtsprechung.

Mit aus diesem Grund liegt den österreichischen ASP Verträgen meist deutsches oder amerikanisches Recht zugrunde [Ciresa 2002]. Die folgende Abhandlung beruht auf deutscher Literatur. Das deutsche Recht ist dem österreichischen im Zusammenhang mit ASP-Verträgen sehr ähnlich, sodass auch in Zukunft, wenn Verträge nach geltendem österreichischem Recht existieren, keine wesentlichen Änderungen der Rechtssituation auftreten werden.

1.7.1. Vertragseinordnung

Eine vertragstypologische Zuordnung ist im Hinblick auf geltendes Gewährleistungsrecht und allgemeine Geschäftsbedingungen trotz der im Zivilrecht geltenden Privatautonomie unbedingt vorzunehmen.

Im Falle einer Lücke im ASP-Vertrag wird auf den Vertragstypus zurückgegriffen, dem der ASP-Vertrag typologisch zuzuordnen ist.

Besonderes Augenmerk liegt hierbei auf den für die Einordnung maßgeblichen **Hauptleistungspflichten**, da eventuell darüber hinausgehende Nebenleistungspflichten nichts am einschlägigen Vertragstypus ändern.

Eine detaillierte Abhandlung, ob der ASP-Vertragstypus als Dienstvertrag, Werkvertrag, Miete, Pacht oder Leasing zu klassifizieren ist, wurde in [Kolk 2001] vorgenommen:

Es zeigt sich, dass ein Dienstvertrag insbesondere dem Interesse des ASP-Kunden nicht entspricht, da der Provider lediglich ein bloßes Tätigwerden schulden würde. Dem Kunden ist aber gerade an der Nutzung der Applikationen, also am Ergebnis dieses Tätigwerdens gelegen. Auch das Werkvertragsrecht ist nicht anwendbar, da der Provider keinen Erfolg im Hinblick auf die Lösung eines betrieblichen Problems schuldet.

Mit der zeitweisen entgeltbehafteten Gebrauchsüberlassung von Software als Hauptzweck, wobei Software als nichtverkörperte Sache klassifiziert wird, ist der ASP-Vertrag als **Mietvertrag** gem. §§535 ff. BGB zu charakterisieren.

Ein Pachtvertrag liegt deshalb nicht vor, weil der Schwerpunkt des Vertrages nicht auf der für die Pacht typischen Fruchtziehung (z.B. wirtschaftliche Verwertung) liegt, sondern auf der bloßen Gebrauchsmöglichkeit bzw. -überlassung von Software.

An Leasing-Varianten scheidet das Finanzierungsleasing aus, da ASP-Software mehrfach überlassen wird und deshalb eine Vertragszielerreichung durch eine Vollamortisierung aufgrund der Leasingraten eines ASP-Kunden nicht gegeben ist und auch der ASP-Kunde nicht am Ende der Vertragslaufzeit Eigentum an der Software erwirbt. Das Operatingleasing jedoch ist auf mehrfache Überlassung der Sache (Software) angelegt. Allerdings ist rechtlich das Operatingleasing ein normaler Mietvertrag.

ASP ist also Miete im Sinne von Gebrauchsüberlassung von Software als Hauptleistung.

1.7.2. Support- und Pflegeleistungen (SLAs)

Eine detaillierte rechtliche Beurteilung aller Zusatzleistungen ist im Rahmen dieser Arbeit nicht möglich. Es wird hier deshalb lediglich auf das Verhältnis der vertraglichen Beurteilung von Zusatzleistungen zum Typus des ASP-Vertrages und die sich daraus ergebenden Konsequenzen im Hinblick auf die anwendbaren Rechtsvorschriften eingegangen.

Als grundlegender Punkt sollte zunächst festgehalten werden, dass sämtliche Zusatzleistungen und SLAs als Nebenpflichten einzuordnen sind und nichts an der Charakterisierung des ASP-Vertrages als Miete ändern. Jede einzelne Zusatzleistung kann sich jedoch nach ihrer eigenen Rechtsnatur richten. So kann eine vereinbarte Datenpflege z.B. als Dienst- oder unter Umständen auch als Werkvertrag anzusehen sein.

Die Einbeziehung von SLAs in den ASP-Vertrag dient vor allem einer Sicherung der Services für den Kunden. Diese können z.B. folgende Inhalte haben: Instandhaltung und Instandsetzung, Beseitigung von Softwarefehlfunktionen, Aktualisierungen (Updates), Systembackups, Verfügbarkeit sowie Kundendienst. Ein weiterer wesentlicher Aspekt, der über SLAs geregelt wird, ist die Frage des Katastrophenmanagements. Es werden diesbezüglich Vereinbarungen über die Anzahl möglicher Ausfälle sowie über Reaktionszeiten bei der Bearbeitung eines Problems getroffen. Empfehlenswert ist eine Einigung über Mean-Time-to-Repair (MTTR) und Mean-Time-to-Failure (MTTF).

Wie die angeführten Inhalte der SLAs zeigen, haben diese erheblichen Einfluss darauf, ob der ASP-Vertrag zur Zufriedenheit der Vertragsparteien ablaufen und deren Erwartungen erfüllen kann. SLAs sollten daher folgende wesentliche Eigenschaften besitzen: Klare Definitionen, Praktikabilität, Beweisbarkeit der Einhaltung bzw. Nichteinhaltung.

1.7.3. Haftung und Gewährleistung

Aus der Einordnung der Online-Softwareüberlassung als Miete ergibt sich unmittelbar der Umfang der gesetzlichen Gewährleistung und Haftung.

Als Vermieter ist der ASP verpflichtet, die Software mängelfrei zu überlassen und während der Vertragslaufzeit in dem vertragsgemäßen Zustand zu erhalten (§ 536 BGB). Diese Erhaltungspflicht umfasst, ohne dass es einer gesonderten Pflegevereinbarung bedarf, auch die Pflicht zur Erhaltung der Gebrauchstauglichkeit der Software sowie zur Instandsetzung und Instandhaltung der Hardware. Während des Mietverhältnisses besteht weiters eine Beratungspflicht gegenüber dem Mieter. Im Rahmen seiner Sorgfalts- und Schutzpflichten hat der Provider dem Kunden Gefahren unverzüglich mitzuteilen (z.B. Verseuchung der Software mit Viren). Diese Pflichten sind formularvertraglich nicht ohne weiteres einschränkbar.

Bei Auftreten von Mängeln hat der ASP-Endkunde einen aus §§ 535, 536 BGB folgenden Erfüllungsanspruch auf Mängelbeseitigung. Zusätzlich ist er für die Dauer der Mangelhaftigkeit zur Minderung des Mietzinses nach § 536 BGB berechtigt. Dies gilt es durch Vereinbarung von Wartungsfenstern bei geplanten "Down-Zeiten" zu vereinbaren. Der Mietzins mindert sich kraft Gesetzes entsprechend dem Grad der Nutzungsbeschränkung. Daneben kann auch ein Schadenersatzanspruch unter den Voraussetzungen des § 536a BGB bestehen. Dieser kommt z.B. bei ungeplanten "Down-Zeiten" zum Tragen. Ebenfalls eine Schadensersatzpflicht des ASPs begründen Mängel der Software, die schon bei Vertragsabschluss bestanden. Für nach Vertragsabschluss auftretende Mängel haftet der ASP nur, wenn er diese zu vertreten hat.

Außerdem hat der Kunde die Möglichkeit, dem Provider eine angemessene Frist für eine Abhilfe gegen Vorenthaltung oder Entzug des vertragsgemäßen Gebrauchs zu setzen, nach deren erfolglosem Verstreichen der Kunde den ASP-Vertrag gem. § 542 1 BGB fristlos kündigen kann, wenn der Gebrauch der Software durch ein ganz besonderes Interesse des Mieters gem. § 542 II BGB gerechtfertigt ist. Wesentlich für das Kündigungsrecht nach § 542 I BGB und § 542 II BGB ist die Tatsache, dass die Möglichkeiten selbst dann bestehen, wenn den ASP an der Vorenthaltung bzw. dem Entzug des vertragsgemäßen Gebrauchs der Software keinerlei Verschulden trifft, was in Anbetracht der großangelegten Partnerschaften bei ASPs von hoher Bedeutung ist. Allerdings ist es mittels Vertragsklauseln möglich, die verschuldensunabhängige Garantiehaftung für anfängliche Mängel, und damit auch die Rechte des Kunden auf Schadenersatzanspruch wegen Nichterfüllung gem. § 538 1 BGB, vertraglich auszuschließen.

Für die schuldhafte Verletzung etwaiger Nebenpflichten (Beratung, Schulung, etc.) haftet der Provider nach den Grundsätzen der positiven Vertragsverletzung. Hiernach hat der Provider dem Kunden den unmittelbar und mittelbar aus der Pflichtverletzung resultierenden Schaden zu ersetzen.

Den Kunden trifft als wesentliche Pflicht aus dem ASP-Vertrag die Zahlungspflicht, welche sich unmittelbar aus § 535 2 BGB ergibt. Eine weitere sich unmittelbar aus dem Mietrecht ergebende Pflicht des Kunden ist die zur Rückgabe der Software nach Beendigung der Mietzeit gem. § 556 1 BGB, welche bei Softwaremietverträgen regelmäßig in eine Pflicht zur Löschung aller vorhandenen Programmkopien umgewandelt wird. Der Nutzer hat weiterhin alles zu unterlassen, was Schaden an der Software verursachen könnte. Überdies sind die Programme vor unberechtigten Zugriffen Dritter zu schützen (§549 Abs. 1 BGB).

Zeigt sich ein Mangel an der Software, ist dieser dem Vermieter (Provider) gem. § 545 1 BGB unverzüglich anzuzeigen. Diese Pflicht gilt jedoch nur eingeschränkt, da der Vermieter neben dem Mieter (Kunden) ebenfalls Zugriff auf die Mietsache (Software) hat. Aus Praxisgesichtspunkten wird man aber dem ASP-Kunden immer raten müssen, den Mangel unverzüglich zur Anzeige zu bringen. Zu den Folgen einzelner Pflichtverletzungen auf Seiten des Kunden (Zahlungspflichtverletzung, Rückgabepflichtverletzung, Überschreitung des vertragsgemäßen Gebrauchs, Verletzung von Nebenpflichten) siehe [Bettinger 2001].

In diesem Papier ist auch ein Verweis auf Vorschläge zu außergerichtlichen Konfliktlösungsverfahren als Methode zur Streiterledigung innerhalb des ASP-Schichtenmodells zu finden.

1.7.4. Lizenzrechtliche Aspekte

Die schuldrechtliche Einordnung der Online-Softwarenutzung als Mietvertrag und ihre Einordnung als urheberrechtliche Verwertungshandlung sind strikt zu trennen.

Das Urheberrecht, also das Recht des Urhebers an seinem individuellen geistigen Werk, wird durch das Urheberrechtsgesetz (UrhG) geschützt. Dieses Recht findet gem. §§ 69 a ff. UrhG auch auf Computerprogramme Anwendung. Wesentlich für das Urherberrecht ist seine Unübertragbarkeit (§ S. 2 UrhG). Um dem Urheber aber die Möglichkeit der Nutzbarmachung seines geistigen Werkes zu ermöglichen, kann er gem. § 31 UrhG einem anderen gewisse Nutzungsrechte einräumen. Diese Nutzungsrechte werden in der Praxis oft **Lizenzen** genannt. Der Kunde erwirbt eine sog. einfache Lizenz, die ihm das Recht gewährt, das Werk neben dem Urheber oder anderen Berechtigten auf die ihm erlaubte Art zu nutzen (§ 31 II UrhG).

Für den ASP stellt sich lizenzrechtlich nun die Frage, welchem speziellen Verwertungsrecht des UrhG diese Lizenz zuzuordnen ist. Wie mehrfach in der Literatur begründet [Kolk 2001][Bettinger 2001], stellt die Gewährung der Online-Nutzung von Software an mehrere Nutzer eine (derzeit noch) unbenannte öffentliche Wiedergabe i.S.d. § 15 Abs. 3 UrhG dar und fällt unter das geplante Online-Übermittlungsrecht. Der ASP muss sich, wenn er die Softwareanwendungen nicht individuell für jeden Kunden auf einem eigenen Server bereithält, das Recht der öffentlichen Wiedergabe (§ 15 Abs. 3 UrhG) sowie das Vervielfältigungsrecht (§ 69c Nr. 1 UrhG) einräumen lassen. Zur Frage, ob dem Kunden durch das Laden der Benutzeroberfläche oder durch den bloßen Programmablauf ebenfalls ein Vervielfältigungsrecht eingeräumt werden muss, siehe [Bettinger 2001].

1.7.5. Datenschutz

Ein weiterer beachtenswerter Aspekt bei dem Abschluss von ASP Verträgen ist der Datenschutz. Datenschutz dient der Sicherung personenbezogener Daten vor Missbrauch (Einsichtnahme, Veränderung, Verwertung unter Beeinträchtigung schutzwürdiger Belange des Betroffenen). Hierzu wurde das Datenschutzgesetz erlassen, dessen Zweck es ist, "den einzelnen davor zu schützen, dass er durch den Umgang mit seinen personenbezogenen Daten in seinem Persönlichkeitsrecht beeinträchtigt wird" (§ 1 1 BDSG). Der Gesetzgeber gibt weiters vor, was unter "personenbezogenen Daten" zu verstehen ist: "Personenbezogene Daten sind Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren natürlichen Person (Betroffener)."

Dem ASP liegen solche Daten vor: Hierzu gehören die Bestandsdaten des Kunden (Name, Adressen, EDV-Kenntnisse, gemietete Software, etc.), die jeweiligen Nutzungsdaten (Welche Software hat der Kunde wann, wofür, wie oft und wie lange in Anspruch genommen ?), etwaige Daten der Arbeitnehmer des Kunden und personenbezogene Daten von Dritten.

Daten von juristischen Personen (GmbH, AG) sind keine personenbezogenen Daten i.S.v. § 3 1 BDSG. Zur Umgehung datenschutzrechtlicher Bestimmungen könnte ein ASP also die Strategie verfolgen, nur Verträge mit juristischen Personen zu schließen, die selbst keine personenbezogenen Daten bearbeiten. Allerdings stellt sich schnell heraus, dass der vertragliche Ausschluss der Datenschutzbestimmungen unpraktikabel ist, da der Provider folglich jeden einzelnen Kunden genau auf das Vorhandensein personenbezogener Daten überprüfen müsste, was sich als unökonomisch erweisen dürfte. Es ist deshalb in der Praxis einfacher, vorsorglich den Datenschutz in sein Geschäftsgebaren mit einzubeziehen.

ASP fällt unter die speziellere Regelung des BDSG, dem **Teledienstedatenschutzgesetz** (TDDSG). Demnach ist es als Teledienst qualifiziert. Mit der Anwendbarkeit des TDDSG ist der Provider als Diensteanbieter i. S. v. § 2 Nr. 1 TDDSG an die dort normierten Pflichten gebunden. So hat er den ASP-Kunden als Nutzer vor Erhebung personenbezogener Daten über Art, Umfang, Ort und die Zwecke der Erhebung, Verarbeitung und Nutzung dieser Daten zu informieren. Diese Information muss für den Nutzer auch jederzeit abrufbar sein. Weiterhin hat der Provider gem. § 4 1 TDDSG den ASP-Dienst so anzubieten, dass dessen Nutung (Nutzungsprofil) und Bezahlung anonym oder unter Verwendung eines Pseudonyms möglich ist. Personenbezogene Daten über die Nutzung sind zu löschen, wenn sie nicht für die Abrechnung benötigt werden (§ 4 II Nr. 1, 2 TDDSG). Schließlich ist noch § 7 TDDSG zu erwähnen, der dem ASP-Kunden ein jederzeitiges unentgeltliches Auskunftsrecht gegenüber dem Provider einräumt.

Bei ausdrücklicher Einwilligung dürfen personenbezogene Daten auch über die Datenschutzbestimmungen hinaus erhoben, verarbeitet und für andere Zwecke verwendet werden (z.B. Marktforschung, Werbung).

Bei Verstößen gegen die Datenschutzbestimmungen drohen theoretisch die in den §§ 43 f. BDSG definierten Straf- und Bußgeldvorschriften (im Höchstfall bis zu zwei Jahre Freiheitsstrafe oder 25.000,- Euro Bußgeld). Im Unterschied zu Ansprüchen von öffentlichen Stellen muss beim Provider jedoch immer ein Verschulden in Form von Vorsatz oder Fahrlässigkeit vorliegen. Das Risiko für den Provider, auf Schadenersatz wegen Verstoßes gegen Datenschutzvorschriften in Anspruch genommen zu werden, ist jedoch ebenso gering wie die Gefahr der Verhängung eines Bußgeldes. In der Praxis sind fast keine Fälle bekannt, in denen eine Verurteilung zur Zahlung von Schadensersatz erfolgte, da im Regelfall ein materieller Schaden beim Anspruchsteller nicht nachgewiesen werden kann und die Gerichte den Ersatz immaterieller Schäden nur äußerst restriktiv anerkennen. Allerdings hat ein durch Rechtsbruch erlangter Wettbewerbsvorteil nicht selten empfindliche Unterlassungs- und Schadenersatzansprüche von Seiten der Konkurrenz bzw. seitens der Interessenverbände zur Folge.

Kapitel 2

Organisation und Technik

In diesem Kapitel werden die organisatorischen und technischen Konzepte beschrieben, die hinter den in Kapitel 1 aufgezeigten Merkmalen von ASP stehen. Dabei wurden die einzusetzenden Mittel und zu betrachtenden Aspekte nach derzeitigen Erkenntnissen aus ASP-Theorie und -Praxis ausgewählt.

Im Kapitel "Software Basistechnologien für ASP" werden vor allem die wichtigsten ASP Features und deren Realisierung beschrieben, während es in den Kapiteln "Server Herausforderungen" und "Client Herausforderungen" eher um qualitätsoptimierende Maßnahmen geht, die aber einen großen Teil der technischen ASP Eigenschaften bestimmen.

Welche Techniken und Organisationsformen letztlich ausgewählt und bevorzugt werden, hängt in hohem Maße von den Anforderungen der zu realisierenden ASP-Lösung ab (damit ist die Art und Zielgruppe der Anwendungen und Dienstleistungen, deren Umfang und Komplexität, deren Vertrieb und Betrieb und deren Vermarktung gemeint). Ein einfacher, kostenlos nutzbarer, ASP-basierter Terminplaner benötigt als Minimalanforderung lediglich Internetfähigkeit und Mehrbenutzerfähigkeit, während eine komplexe ERP-Lösung (mit "Managed Services", siehe voriges Kapitel) höchste Ansprüche an alle beschriebenen Techniken stellt.

Somit wird klar, dass man sowohl nach Arten von ASP-Lösungen als auch nach den allgemeinen technischen Voraussetzungen und Realisierungen einteilen könnte. In dieser Diplomarbeit wurde letztere Variante gewählt, da das Angebot an Arten von ASP-Lösungen mittlererweile unüberschaubar ist. Die gemeinsame Basis für diese Abhandlung ist ASP in der Ursprungsbedeutung (siehe auch Kapitel 1.3.3., "Technologiesicht"), und bezieht sich auf eine "ASP-Musterlösung": **Benutzen** von **Anwendungen** über **Internet** (oder ein privates Netzwerk) auf **Mietbasis**.

2.1. Software Basistechnologien für ASP

In diesem Kapitel werden technische Prinzipien und Technologien aus der Praxis erklärt, die unterstützen, ASP-fähige Software neu zu entwerfen und zu implementieren oder herkömmliche Software an ASP anzupassen.

Dabei stellt sich die Frage, auf welcher Ebene man die ASP Features realisiert. Derzeit gibt es dafür zwei unterschiedliche technische Ansätze, aber es ist noch nicht absehbar, wie vielschichtig oder monopolistisch ASP Technologien in Zukunft sein werden.

2.1.1. Terminalprogramme

Die erste Lösung ist, die ASP Features auf einer höheren Ebene als die Applikationsebene mittels sogenannter Terminalprogramme zu realisieren. Terminals erlauben es, Betriebsystemsessions und/oder Applikationen eines Rechners von einem anderen Rechner aus

quasi fernzubedienen. Das Prinzip ist einfach: Textuelle und graphische Ausgaben der dem Terminalprogramm untergeordneten Applikation (die vollständig am Server läuft) werden über das Internet an einen Client umgeleitet, der die Informationen lediglich visualisieren muss. Umgekehrt wandern Tastatur- und Mauseingaben vom Client zum Server über die Leitung und werden an das jeweilige Programm weitergeleitet. Dabei sind meist spezielle Kommunikationsprotokolle (in erster Linie über TCP/IP) im Einsatz, die eigene IP-Ports benutzen.

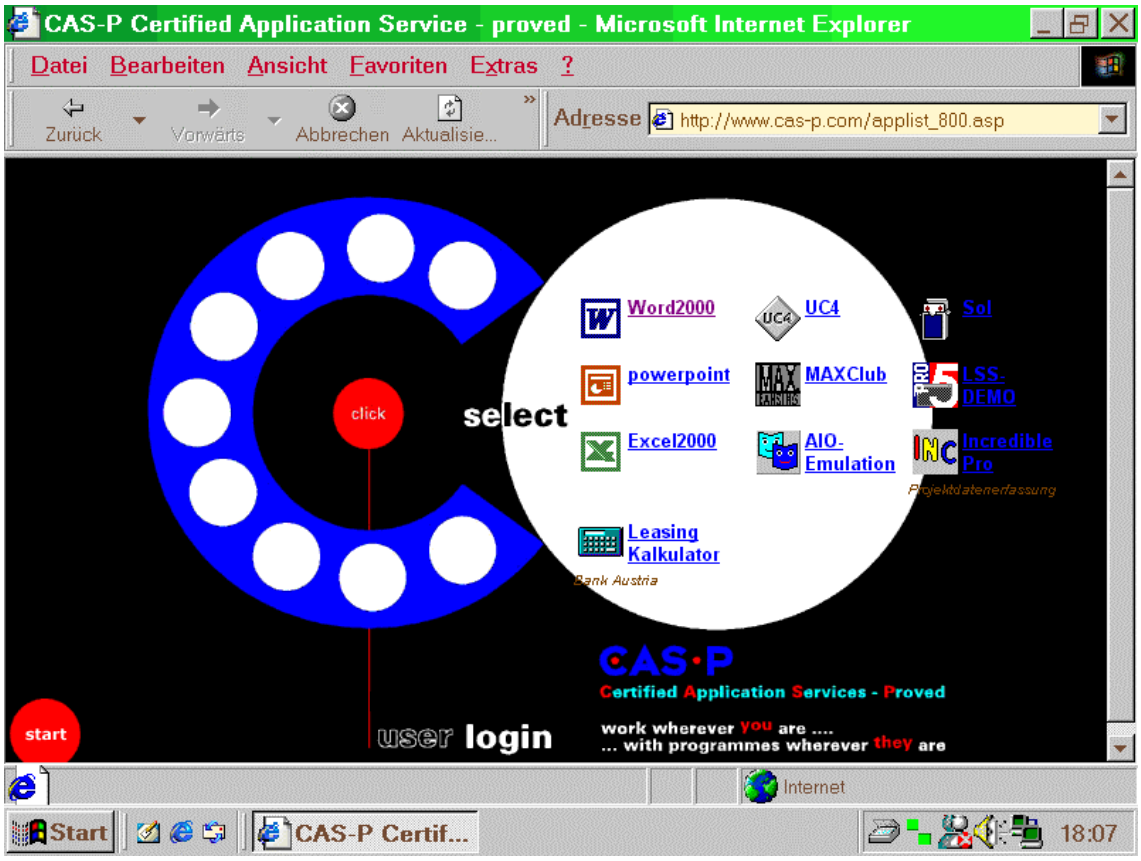

Abbildung 2-1 : Portal für Internetapplikationen auf Terminalbasis: http://www.cas-p.com

An den Applikationen muss in der Regel nichts oder nur sehr wenig geändert werden, um sie auf diese Weise fernbedienbar zu machen. Auf Client und Server laufen jeweils Kontrollprogramme, die die Daten entsprechend umleiten. Kontrollprogramme für den Client gibt es als eigenständige Programme oder für die gängigsten Internetbrowser (Netscape, Internet Explorer) z.B. in Form eines Plug-Ins (Netscape), einer ActiveX-Komponente (Internet Explorer) oder als Java-Applet. Dies erlaubt ein gewisses Maß an Geräteunabhängigkeit, an das Look & Feel der serverseitigen Betriebsystemplattform bleibt man aber gebunden. Dadurch wird aber nicht nur Zugang zu einer Applikation geschaffen, sondern man profitiert prinzipiell von allen Funktionen des serverseitigen Betriebsystems wie z.B. von Multitasking, Virenschutz, Internet, Konfigurierbarkeit des Systems und insbesondere vom **Dateisystem**. Dieses ist meist so eingerichtet, dass für jeden Benutzer oder für jede Benutzergruppe eigene Zugriffsrechte auf die Dateien und Verzeichnisse des Servers existieren.

Da nicht HTTP als Kommunikationsprotokoll verwendet wird, bleibt die Internetseite bzw. -adresse während der gesamten Laufzeit des browserfähigen Clientkontrollprogramms unverändert. ASP über Terminals ist **sessionbasiert**. Eine Session beginnt entweder vor oder

nach der Authentifizierung und dauert bis zu einem Timeout, dem Abmelden oder Schließen des Clientkontrollprogramms oder bis zum Wechsel der Internetadresse.

Graphische Terminalprogramme gibt es inzwischen für viele Systeme, manche sind auf ASP optimiert. Hier zu nennen wären z.B. MS Windows Terminal Services (WTS), Citrix Metaframe, SCO Tarantella und X Windows für UNIX. Die am meisten verbreitetsten Produkte sind allerdings WTS und MetaFrame. WTS unterstützt serverseitig nur die Windows NT-Plattform (NT 4.0, Windows 2000). MetaFrame wird auch für UNIX angeboten. Etablierte Software wie z.B. Microsoft Office, PowerPoint oder SAP wird fast ausschließlich über WTS oder Citrix MetaFrame angeboten.

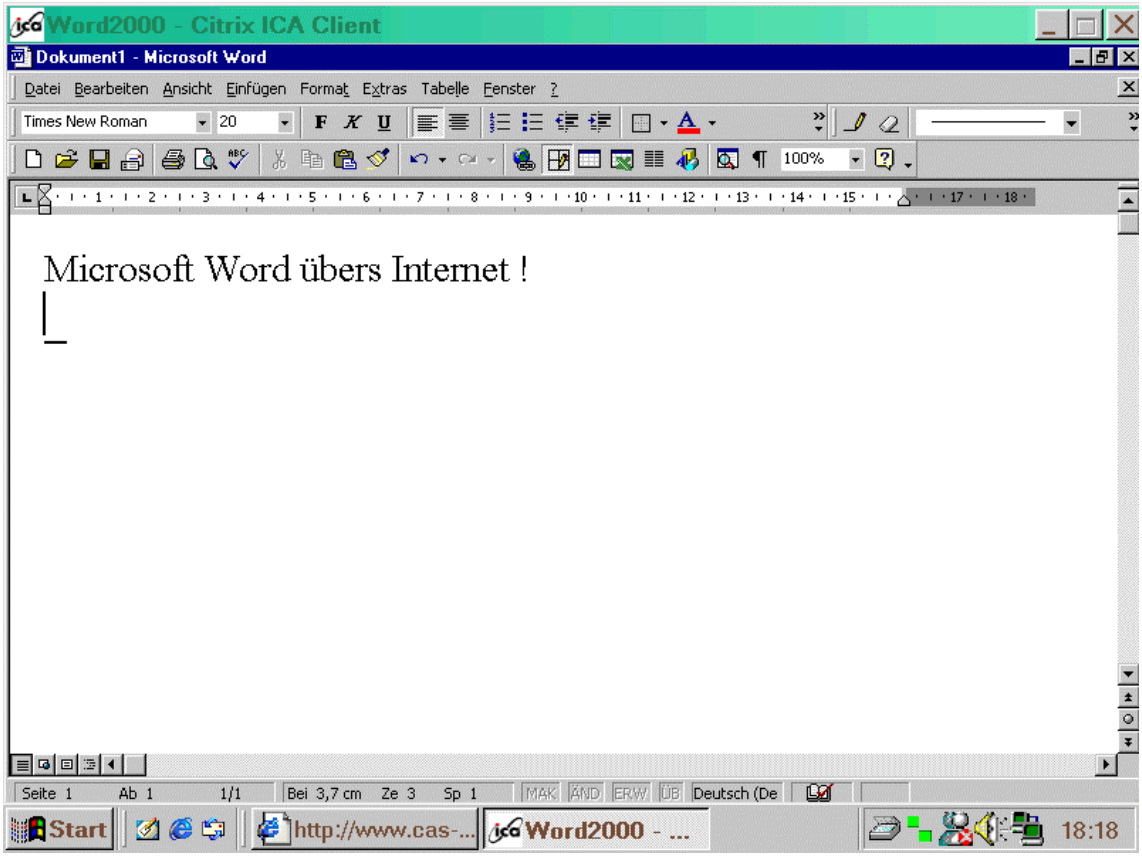

Abbildung 2-2 : Microsoft Word über den browserfähigen Citrix-Terminalclient

Terminalprodukte werden ausführlich in Kapitel 3.4. und 3.5. beschrieben.

Auf http://www.runaware.com, http://www.cas-p.com und http://www.tuul.de lassen sich einige Applikationen über die Terminalbasistechnologie kostenlos über das Web testen.

2.1.2. Websprachen

Die zweite Lösung ist, die ASP Funktionen direkt in eine Applikation einzuprogrammieren, sei es durch Neuerstellung einer Applikation oder durch Umprogrammieren. Hierbei gibt es Konzepte (z.B. CGI, Enterprise Java Beans (EJB)), Programmier- bzw. Websprachen (Java Servlets, Java Applets, JavaScript, VBScript, PHP) und Plattformen (z.B. Microsoft .NET, BEA WebLogic, IBM WebSphere), die dabei helfen. Die gebräuchlichste Methode dabei ist, server- und clientseitige Websprachen dafür einzusetzen. Als Laufzeitumgebung für den Endbenutzer dient in diesem Fall fast immer ein Internetbrowser, der über HTTP mit dem Server kommuniziert, selten ist eine eigene Laufzeitumgebung erforderlich, denn die würde die Orts-

und Geräteunabhängigkeit von ASP beeinträchtigen. Zur Demonstration ein Beispiel einer webbasierten Applikation aus Anwendersicht:

Oracle bietet auf http://crmonline.oracle.com die CRM (Customer Relationship Management)-Produkte **Support.Oracle.com** und **Sales.Oracle.com** nach erfolgreicher Registrierung zum kostenlosen Test an:

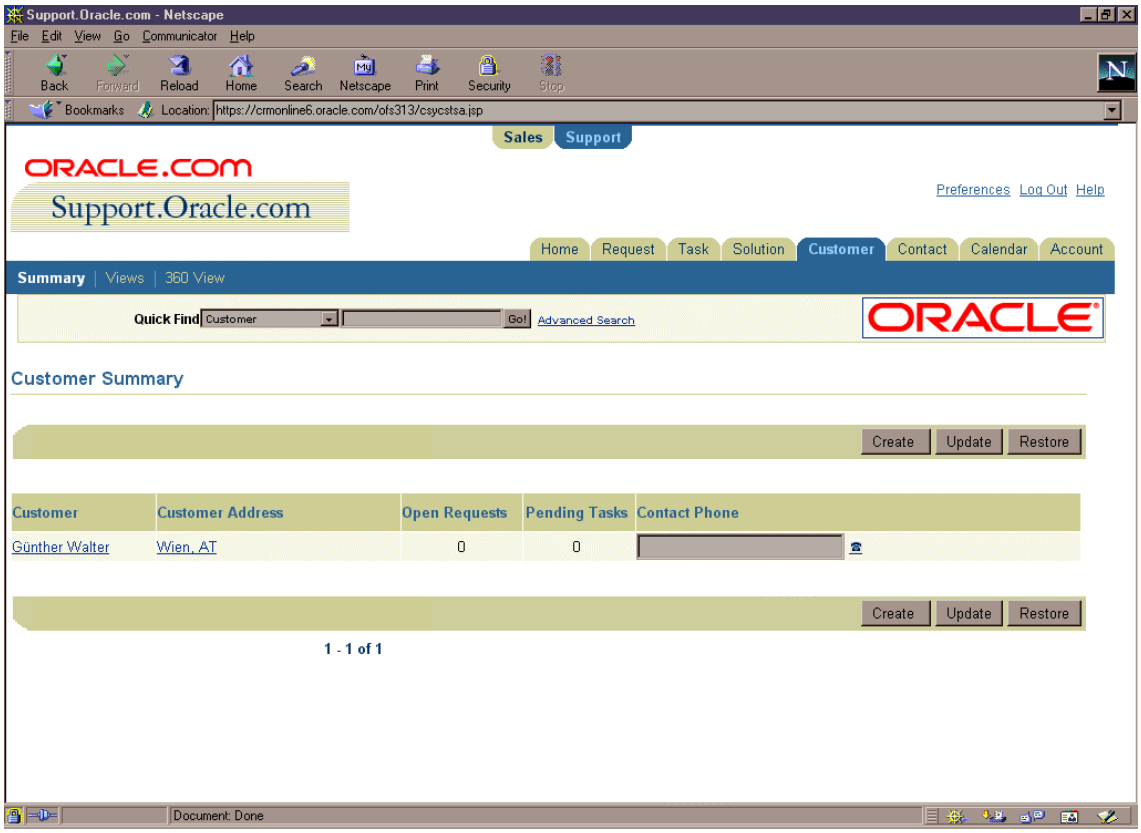

Abbildung 2-3 : Eine websprachenbasierte ASP-Applikation aus Anwendersicht: Support.Oracle.com : http://crmonline.oracle.com

Es handelt sich dabei um Applikationen, mit deren Hilfe es Unternehmen möglich ist, Verkäufe und Kundeninteressen zu verfolgen und auf Basis der Software-Reports Unternehmensstrategien zu setzen. Die Lösung basiert serverseitig auf Java Server Pages und clientseitig ausschließlich auf HTML und JavaScript. Anhand dieses Beispiels sollen später in der Diplomarbeit noch einmal die Eigenschaften von websprachenbasierten Applikationen aus Benutzersicht dargestellt werden.

Der große Unterschied zum Terminalansatz ist, dass nicht nur grafische Daten übertragen werden, sondern auch ein gewisses Maß an "Intelligenz" (in Form von Browsersprachen wie HTML, Javascript, Java, VBScript usw.). Um beim ASP-Konzept zu bleiben und die technische Umsetzung und Wartung nicht unnötig zu verkomplizieren, dient dieser Client-Code im Regelfall nur zur Verarbeitung, Visualisierung und Weiterleitung der Ein-/Ausgabe des Servers.

Durch die erforderliche Lauffähigkeit im Browser und die Einschränkung auf das HTTP-Protokoll resultieren Schwierigkeiten beim Umschreiben von Software auf ASP, nämlich die der **Sessionlosigkeit** von HTTP und der **Seitenorientierung** von Browsern. Diese wesentlichen Eigenschaften sollen kurz erklärt werden:

Eine HTTP-"GET"-Anforderung bzw. Session, sie beginnt z.B. durch Eingabe einer Internetadresse oder Drücken eines Buttons, liefert als Ergebnis immer eine Internetseite, die der Browser visualisiert, damit ist die HTTP-Session abgeschlossen. Jede Änderung der Ausgabe oder eines Teils der Benutzerschnittstelle erfordert die Neuübertragung der gesamten

Seite oder des Frames. Um Interaktivität in einer solchen seitenorientierten Umgebung zu erreichen, müssen entweder alle möglichen Resultate, zusammen mit Intelligenz in Form einer Browsersprache, im voraus übertragen werden, oder die Anforderung wird "URL-codiert"[1].

Ein zweites Problem ergibt sich durch die Sessionlosigkeit: Ohne gesonderte Maßnahmen existiert nach jeder abgeschlossenen HTTP Anforderung keine Verbindung mehr zum Server. Bei der Arbeit mit Applikationen werden aber viele Arbeitsschritte mit wechselnden Zuständen getätigt. Eine gebräuchliche Methode ist, diese Zustände am Client in "Hidden-Forms" oder "Cookies" zu speichern und bei jeder Serverinteraktion "URL-codiert" mitzuschicken (Client-Sessionvariablen). Als Alternative dazu oder zusätzlich kann man Zustände auch serverseitig (in Verbindung mit einer eindeutigen Benutzer-ID) speichern. Solche Server-Sessionvariablen werden entweder auf Befehl des Clients angelegt, verändert bzw. gelöscht oder die Session wird durch ein Server-Timeout abgebrochen.

Diesen Problemen wird versucht, durch Maßnahmen auf anderen Ebenen beizukommen. Die Anzahl der Internetseiten läßt sich z.B. durch ein intelligentes Java-Applet auf eine einzige reduzieren. Dann wird jedoch der Großteil der Applikation lokal ausgeführt, was dem ASP-Konzept widersprechen würde. Diese Technik ist außerdem bei großen Applikationen wegen der langen Downloadzeiten nicht durchführbar.

Manche fortschrittlichen Programmiersprachen und Plattformen zur ASP-Softwareentwicklung (z.B. Microsoft .NET, IBM WebSphere und BEA WebLogic) trennen nicht mehr strikt zwischen Server- und Client-Code, sondern zwischen Benutzerinterface und Applikationsprogrammlogik, der Client-Code für den Internetbrowser wird dabei automatisch erzeugt, ebenso wie Code zum Verwalten von Sessioninformation. Dies entlastet den Programmierer, da kein Code mehr für die Weiterleitung von Ein- und Ausgabe zwischen Client und Server implementiert werden muss und erleichtert das Umschreiben vorhandener Software für ASP. Trotzdem "entkommt" man der Eigenschaft der Seitenorientierung nur zum Teil: Dieses Problem und der Aspekt der Codegenerierung werden näher im Kapitel "Microsoft .NET" im praktischen Teil dieser Diplomarbeit behandelt.

Auch serverseitig gibt es z.Z. Unterschiede zum Terminal: Im Gegensatz zur Terminaltechnologie wird aus Sicht des Client fast immer auf eine Anbindung zum Serverdateisystem verzichtet. Damit geht man zahlreichen Schwierigkeiten, vor allem jene die Performance betreffend, aus dem Weg. Um z.B. eine Dateiauswahlbox zu implementieren benötigt man ausreichend Client Intelligenz (Java) und bei der Verwendung müssen Nachrichten mit umfangreichen Informationen (Dateinamen, Attribute, Piktogramme) übertragen werden. Bei einer grafischen Lösung wie dem Terminal sind solche semantischen Informationen dem Client gar nicht bekannt.

Stattdessen sind webbasierte Programme an **Datenbanken** angebunden. Eine Anbindung an eine Datenbank ist sowohl server- als auch clientseitig wesentlich performanter, insbesondere bei wachsender Benutzeranzahl. Zum zweiten wird bei Websprachen, wie bereits erwähnt, Sessioninformation gespeichert. Diese ist mit einer Datenbank wesentlich einfacher zu handhaben [Paiha 2001]. Eine Datenbank bietet Transaktionalität (d.h. die Datenbank lässt sich im Fall eines Fehlers immer in einen konsistenten Zustand bringen) und ist besser skalierbar und replizierbar als ein Dateisystem. Allerdings sind ohne Dateisystem die Möglichkeiten, je nach Anwendung, eingeschränkt.

[1] z.B. in der Form "http://meineperson.com?name=Walter"

2.1.3. Vergleich von Terminals und Websprachen

Ein Vergleich von terminalbasierten Systemen und Websprachen in ASP-Theorie und -Praxis erfolgt an dieser Stelle; bei der Diskussion der Anforderungen an ASP Basistechnologien ab Kapitel 2.2 werden aber immer wieder neue Aspekte dieser beiden Technologien genannt:

Ein möglicher Vorteil der Methode mit Websprachen wäre, dass durch die theoretisch beliebig erweiterbare Intelligenz des Clients dieser flexibler und besser mit Fremdapplikationen integrierbar wäre. Durch diese Intelligenz ist es auch möglich, von der Applikation aus lokal vorhandene Hardware besser zu unterstützen (z.B. Scanner, Hardwareschnittstellen wie USB etc.). Bei der Terminaltechnologie ist dies zwar auch möglich (wie dies geschieht, wird später erklärt), man ist aber immer auf die Möglichkeiten des clientseitigen Kontrollprogramms beschränkt, welches sich wegen der "fehlenden Intelligenz" in den Kommunikationsdaten nicht an die spezifischen Anforderungen jeder einzelnen Applikation anpassen kann.

Der Aufwand für eine Realisierung von ASP durch Websprachen und -Plattformen ist jedoch wegen der notwendigen Um- bzw. Neuprogrammierung höher als beim Terminal, da dort in den meisten Fällen lediglich der serverseitige Teil des Terminalprogramms konfiguriert werden muss und die Applikation unverändert bleiben kann, sofern die gewünschte Applikation schon vorhanden ist. Weiters einige Schlussfolgerungen aus [Paiha 2001]:

ASP über Terminals, so scheint es, ist die schnellste und gleichzeitig leistungsfähigste Möglichkeit, Software ASP-basiert auf den Markt zu bringen. Nachteile sind die hohen Anforderungen an Server-Ressourcen bei wachsender Benutzerzahl (kritischer Wert: ab ca. 30 Benutzer), die schlechte Skalierbarkeit und die hohen Kosten, um Performance und Hochverfügbarkeit zu erreichen. Die Anforderungen an Netzqualität, insbesondere die der Antwortzeit, sind höher. Genaue Zahlen werden später im praktischen Teil genannt.

Applikationen auf Websprachenbasis müssen zwar erst einmal aufwendig extra implementiert werden, der Betrieb ist jedoch wesentlich günstiger. Ein gutes Beispiel sind die weit verbreiteten sog. "Web-Shops" und Messaging-Lösungen wie "Web-Mail" u.ä. Sie stellen das Hauptangebot an websprachenbasierten ASP (im seriösen Sinne mit SLAs und Hochverfügbarkeit) dar. Diese Anwendungen sind für eine Anbindung an eine Datenbank prädestiniert. Die Datenbank skaliert sehr gut (100-1000 Benutzer) und ist einfacher zu handhaben und zu warten als ein Dateisystem, insbesondere im Fehlerfall (des Clients oder des Servers). Das Benutzerinterface ist einfach, es ist wenig Interaktivität erforderlich, daher sind keine hohen Anforderungen an die Antwortzeit des Netzes gestellt.

Das Load-Balancing, also das Aufteilen von Rechenleistung auf mehrere Rechner bzw. Prozessoren, ist mit websprachenbasierten Systemen wesentlich einfacher zu bewerkstelligen.

Alle Applikationen, die aus irgendwelchen Gründen eine Datenschnittstelle zum Clientgerät oder zu anderen Client-Applikationen (z.B. via XML) brauchen (Downloads), sind mit Websprachen einfacher zu realisieren.

Auf der anderen Seite sind terminalbasierte Dienste auf Client-Seite wesentlich plattformunabhängiger als Websprachen (Begründung folgt später). Die clientseitige Wartung ist jedoch, wie man später noch sehen wird, durch explizite Updates etwas aufwendiger.

Wie bereits erwähnt, ist es äußerst aufwendig, im strengen Sinn von ASP teilweise sogar unmöglich, größere Anwendungen mit Websprachen zu realisieren.

Deshalb sind die Terminaldienste im ASP Bereich derzeit führend. Noch ist aber völlig unklar, welche Methode als Sieger hervorgehen wird. Neue Verfahren werden hinzukommen: Vorstellbar wären z.B. innovative Compiler, die Programmquellcode bisheriger, herkömmlicher Software automatisch für eine bestimmte ASP-Plattform kompilieren können, oder speziell für ASP entwickelte Browser (ein einfaches Beispiel dafür gibt es schon: Microsoft MSN Explorer [MSN 2002]).

Mit dem Bestreben, das Internet ASP-fähig zu machen, ist auf dem Gebiet der Software Basistechnologien derzeit eine starke Weiterentwicklung in Gange.

Im folgenden werden die Anforderungen an diese Systeme genannt.

2.2. Internetfähigkeit und Kommunikation

Internetfähigkeit ist eine elementare Eigenschaft jeder ASP Lösung. Als Architektur für die kommunizierenden Einheiten wird für ASP das 3-Schichten-Modell (Client-Server-Datenbank bzw. -Dateisystem) bevorzugt. Voraussetzung für Internetfähigkeit einer ASP-Lösung ist natürlich das Vorhandensein von Internetinfrastruktur: Eine schnelle, sichere und ausfallssichere Internetleitung, Clientinfrastruktur und Serverinfrastruktur mit geeigneter Firewall- und Proxykonfiguration herstellen; das alles sind Aufgaben des Network Service Providers. Eigenschaften dieser Komponenten, die speziell von ASP gefordert werden, werden später gesondert besprochen.

Kommunikationsschnittstellen im 3-Schichten-Modell sind nun notwendig zwischen Client und Server (*Front-End Kommunikation*), zwischen Server und anderen Servern oder Komponenten im Intra- oder Internet (*Middleware*) und zwischen Servern und Datenbanken bzw. Dateisystemen (*Back-End Kommunikation*).

2.2.1. Front-End Kommunikation

In diesem Kapitel wird beschrieben, wie man eine Applikation an das Internet und die Client-Infrastruktur anbindet.

Im Falle der **Terminaltechnologie** wird diese Funktion des Internetzugriffs vollständig vom übergeordneten Terminalprogramm erfüllt und man braucht sich deshalb auf Applikationsdesignebene keine Gedanken zu machen.

Im Prinzip funktioniert das Messaging bei grafischen Terminals so: Beim Verbindungsaufbau wird, sofern von der Terminalsoftware vorgesehen, der Server mit der geringsten Last ausgewählt. Dieses Verfahren wird "Load Balancing" genannt. Danach ist eine direkte Verbindung (Session) zu einem dezidierten Server aktiv. Am Server wird ein spezieller, virtueller Bildschirmtreiber installiert und aktiviert. Beim Beschreiben dieses virtuellen Bildschirmspeichers gibt es mehrere Möglichkeiten: Die GUI-Elemente (Buttons, Menus, Icons, etc.) können z.B. einzeln mit ihren logischen Attributen (Koordinaten, Farben, Stil etc.) eingetragen werden (in X-Windows für UNIX wurde es so realisiert). Eine andere Technik besteht darin, die Differenz aus der Bitmap-Information des Gesamtbildschirms zum vorigen Bildschirm zu speichern (in WTS und Citrix ist sie so implementiert). Wie auch immer die Grafikinformation codiert wird, sie wird von der Nachrichten-Engine konsumiert und über einen speziellen Port des darunterliegenden Nachrichtenprotokolls (z.B. TCP, UDP, IPX, SPX, NetBIOS) zum Client übertragen. Umgekehrt sendet der Client Eingabedaten (Maus, Tastatur) ebenfalls über einen speziellen Port zum Server. Dieser Port muss sowohl auf Client- als auch auf Serverseite bei den betreffenden Firewalls geöffnet sein. Die Datenpakete können weiters in komprimierter und/oder verschlüsselter Form übertragen werden. Die für eine Internetanbindung an eine Applikation notwendige Trennung von Applikationslogik und Benutzerinterfacelogik geht bei Terminals zur Laufzeit vonstatten.

Bei **Websprachen** bzw. Webplattformen sieht es anders aus: Es sind bereits zum Zeitpunkt des Applikationsdesigns Entscheidungen zu treffen, über welches Protokoll Client und Server kommunizieren (HTTP oder ein spezielles Protokoll, verschlüsselte (HTTPS) oder unverschlüsselte (HTTP) Übertragung, verbindungsorientiertes oder verbindungsloses Protokoll) und welche Hilfsprogramme und Plug-Ins dafür notwendig sind.

Die Internetanbindung wird erreicht, indem jedes Client-Ereignis (Mausklicks wie z.B. auf einen Button, Eingabeabschluss durch „Enter“, usw.) mit einer Internetanforderung (bei HTTP in Form einer HTTP-GET oder HTTP-POST Anforderung via einer URL) an den Server verknüpft wird. Dies erfordert aber eine Trennung von Benutzerinterface Programmlogik und Applikationsprogrammlogik zur Design- und Implementierphase. Als Resultat der Internetanbindung und Verteilung einer ASP-Anwendung werden unter "Server Herausforderungen" und "Client Herausforderungen" diese beiden Logiken getrennt behandelt.

2.2.2. Middleware

Manche ASP-Applikationen (Supply Chain Management, E-Commerce) sind verteilte Anwendungen und benötigen zusätzlich zur Verbindung zum Client eine Intranet- oder Internet-Schnittstelle zu anderen Geschäftsobjekten. Zudem beruhen Applikationen zunehmend auf einer verteilten, komponentenbasierten Architektur, um bessere Skalierbarkeit, Performance, Sicherheit und Interoperabilität zu erreichen.

Zu diesem Zweck dienen die sog. **Komponentenmodelle** und **Interprozesskommunikationsprotokolle** (COM, COM+, DCOM, CORBA, EJB mit Java RMI, SOAP), die zusammen eine sog. *Middleware* formen. Die Middleware ist serverseitig angesiedelt und dient der Anbindung an im Netz verteilten Ressourcen an die Internet-Anwendung. In der Praxis existiert nicht immer eine klare Trennung zwischen Komponentenmodellen und Interprozesskommunikationsprotokollen. Server-Produkte, die eben genannte Technologien integrieren, werden auch oft **Application Server** genannt. Diese konkreten Technologien werden im praktischen Teil unter "Traditionelle Internettechnologien und -sprachen" kurz angeschnitten. Drei Beispiele von Application Servern (Microsoft .NET, BEA WebLogic und IBM WebSphere) werden dort ebenfalls vorgestellt.

Interprozesskommunikationsprotokolle (RMI, SOAP, Protokoll-Spezifikation von DCOM und CORBA) sind entweder für den Einsatz im Intranet optimiert (RMI, DCOM) und verwenden daher spezielle TCP/IP-Ports oder eignen sich durch Verwendung des HTTP-Protokolls (SOAP) auch für das Internet.

Komponentenmodelle (COM, COM+, Komponenten-Spezifikation von DCOM und CORBA3) unterstützen die Verwaltung von verteilten Komponenten mit Transaktions-, Sicherheits-, Skalierbarkeits-, Verteilungs- und Installationsdiensten.
Der Einsatz von Middleware ist nicht nur auf websprachenbasierte Applikationen beschränkt, sondern funktioniert auch mit herkömmlichen Applikationen und Applikationen auf Terminalbasis.

2.2.3. Back-End Kommunikation

Im Back-End-Bereich befinden sich Einrichtungen zur Datenverwaltung (Datenbanken, ERP-Systeme, Dateisysteme).

Datenbanken stellen eine Möglichkeit dar, hohe Datenmengen zu speichern, zu strukturieren und zu suchen. Sie bieten zudem bessere Skalierbarkeit und Wartbarkeit als Dateiserver. Die Entscheidung, ob ein Dateisystem oder eine Datenbank die bessere Möglichkeit zur Anbindung an Daten darstellt, ist applikationsabhängig, aber auch direkt von der verwendeten Basistechnologie abhängig. Während Terminalsysteme über Zwischenstufen (dem Terminalserver und dem Betriebsystem) eine mehr oder weniger direkte Anbindung des Clients an das **Serverdateisystem** ermöglichen, unterstützen Serverarchitekturen auf Websprachenbasis eine solche Anbindung nur unzureichend.

Der Einsatz von Datenbanken hingegen ist nicht nur auf websprachenbasierte Applikationen beschränkt, sondern funktioniert auch mit herkömmlichen Applikationen und Applikationen auf

Terminalbasis. Die Datenverwaltung von Internetapplikationen über Datenbanken ist nicht erst seit ASP eine sehr ausgereifte Sache.

Zur Anbindung an Datenbanken gibt es spezielle Treiber (ODBC, JDBC) und Sprachen (SQL). Die wichtigsten Internetsprachen haben eine Datenbankschnittstelle integriert.

2.3. Mehrbenutzerfähigkeit und Zugriffsverwaltung

2.3.1. Mehrbenutzerfähigkeit

Mehrbenutzerfähigkeit nennt man jene Eigenschaft von ASP Basistechnologien, eine Applikation gleichzeitig mit personalisierten Daten und Einstellungen mehreren Benutzern zur Verfügung zu stellen.

Je nach Anwendung gibt es Einrichtungen zur gleichzeitigen Programmausführung, zur Anonymidentifizierung von Benutzern oder zur Authentifizierung, zur Speicherung von Benutzerdaten, Konfigurationen und Dateien in eine Datenbank oder in ein Dateisystem.

Ein wichtiger, kostensparender Aspekt von ASP ist, dass sich die Benutzer die für diese Einrichtungen notwendigen Ressourcen teilen. Im Idealfall soll der Benutzer von der Mitbenutzung der Ressourcen durch andere nichts merken. Jedem Benutzer soll eine eigenständige Umgebung von Ressourcen suggeriert werden, obwohl die Ressourcen physikalisch von mehreren Anwendern mitbenutzt werden.

Auf unterster Ebene ist dies die Rechenleistung: Betrachtet man die gleichzeitige Programmausführung für mehrere Benutzer auf einem Prozessor, ist das zugrundeliegende Serverbetriebsystem dafür verantwortlich. An dem Punkt, wo die Internetanbindung, d.h. die Aufspaltung der Daten wie z.B. Ein- und Ausgabe auf die verschiedenen Benutzer und die Datenübertragung geschieht (das sog. *Delivery*) treten jedoch andere Komponenten in Kraft:

Diese Funktion kann entweder im Webserver (z.B. Microsoft IIS, Apache Webserver) integriert sein, wie es bei den Websprachen und manchen Terminals (Tarantella) der Fall ist, oder durch einen eigenen sog. *Application Server* zur Verfügung gestellt werden. Beispiele für letztere Variante sind: Terminals wie Citrix und WTS, Webplattformen "BEA Weblogic" oder "IBM WebSphere".

Es gibt ASP Applikationen, die schon mit eben besprochener Minimalfunktionalität betreff Multiuserfähigkeit auskommen: Eine kostenlose ASP Applikation ohne Speicherung von individuellen Benutzerdaten kann ohne weiteres auch ohne Authentifizierung (d.h. ohne Log-In) auskommen (z.B. ein einfaches Online-Spiel oder ein Sprachenübersetzungsdienst wie z.B. "Babelfish").

Einen Schritt weiter geht folgende Methode: Anonymidentifikation von Benutzern bedeutet eine eindeutige 1:1 Zuordnung zwischen einer Applikationssitzung und einem Benutzer mittels einer (öffentlichen oder geheimen) Identifikationsnummer. Im einfachsten Fall ist dies die IP Adresse des Endgeräts. Im Gegensatz zum sessionlosen "surfen" hat das Benutzen eines Programms über das Internet einen definierten Anfang, eine Reihe von Transaktionen wie Eingabedaten, Berechnungen und Ausgabedaten und ein definiertes Ende. Dies macht zusätzlich die Vergabe einer Session- oder Benutzer-ID bei mehr als einer Transaktion pro Programmausführung notwendig (siehe auch Kapitel 2.1.2., "Websprachen"). Durch anonyme Benutzer-IDs ist der Server in der Lage, jedem Benutzer während einer Programmsession ein "Gedächtnis" der getätigten Transaktionen zur Verfügung zu stellen. Mit diesem Hilfsmittel ist die Mehrbenutzerfähigkeit schon ausgeprägter als durch bloße Identifikation durch die IP-Adresse mittels dem Webserver. Durch diese Methode werden zwar personalisierte Daten gespeichert, aber nur für jeweils eine Programmsession, und nicht dauerhaft für weitere.

Zusätzlicher Aufwand kommt hinzu, wenn jedem Benutzer dauerhaft private Ressourcen und persönliche Voreinstellungen zur Verfügung gestellt werden, die serverseitig gespeichert werden. Man spricht hierbei von "Persistenz von Daten". Beide Technologien (Terminals und Websprachen) stellen Server-Ressourcen wie z.B. Programmfunktionalität und Rechenleistung zur Verfügung. Die zentrale "persönliche" Serverressource ist beim Terminal aber das private Benutzerverzeichnis im Serverdateisystem, bei den Websprachen sind es benutzerspezifische Einträge in einer Datenbank.

Zum persönlichen Benutzerprofil zählt weiters:

- Benutzerdaten: Netzzugang und Applikationszugang (Einwahlnummern, Verbindungsparameter zum Gateway, Username, Passwort) , Vor und Zuname des Benutzers, Digitale Unterschrift (Zertifikate), Zugriffsberechtigungen, Softwarelizenzinformation, Kostenaufstellungen, etc.
- Protokolle: Logdateien über Applikationennutzung, Statistiken
- System- und Programmkonfigurationen (wie z.B. Farben, Schriftgrößen, Kommunikationsoptionen)
- Sonstige Parameter (Maximaler Benutzerspeicherplatz)

Wenn eine Applikation selbst nicht multiuserfähig ist und nicht speziell für ASP programmiert wurde, wie es beim Einsatz von Terminals der Fall ist, ergibt sich das Problem, dass aus technischer Sicht Programmeinstellungen schwieriger zu personalisieren sind. Eine Programmeinstellungsdatei wird manchmal nicht in einem Verzeichnis für gemeinsame Dateien oder in dem Benutzerverzeichnis abgelegt, sondern im Programmverzeichnis. Dieses ist jedoch physikalisch, genauso wie die Applikation selbst, nur einmal vorhanden. Bei der Benutzung einer Internet-Applikation von mehreren Benutzern gleichzeitig werden lediglich mehrere Instanzen des physikalisch nur einmal gespeicherten Programms geladen und gestartet, welche aber nicht dafür ausgelegt sind, individuelle Programmeinstellungsdateien zu laden. Deshalb müssen solche Programme bei der Umstellung auf ASP gepatcht werden. Man spricht dabei von "die Applikation ASP-ready machen". Alle anderen zum Benutzerprofil gehörenden Daten können vom Terminal verwaltet werden, da sie nicht direkt zur Applikation gehören. Laut [Paiha 2001] sind bereits die meisten Applikationen ASP-Ready, vor allem die Microsoft-Produkte (MS Office etc.).
Es gibt zwei Lösungen, um nicht-multiuserfähige Applikationen ASP-tauglich zu machen:

1. Der Softwarehersteller macht die Applikation multiuserfähig (durch Implementieren von dafür notwendiger Funktionalität)
2. Der ASP stellt für diese Art von Anwendung für jeden Benutzer einen dezidierten Server oder zumindest eine dezidierte Applikation zur Verfügung.

Letztere Variante erscheint selten sinnvoll, da das ASP Modell auf "economies of scale", also Kostenvorteile im Betrieb durch eine Vielzahl an Benutzern, aufbaut.

2.3.2. Zugriffsverwaltung

Es gibt viele Motivationen, Zugriffsverwaltungsfunktionen in ASP Software zu integrieren: Alle resultieren aus der zentralen Verwaltung und der Verwendung von gemeinsamen Ressourcen von mehreren Benutzern.

Die wichtigsten sind:

- Schutz und Geheimhaltung von persönlichen Daten
- Bedarf nach individuellen Konfigurationen und Programmeinstellungen (siehe Kapitel 2.3.1.)
- Erfordernis von individuellen Berechtigungen zum Ausführen von lizenzierten Applikationen (Lizenzverwaltung)
- Eindeutige Zuordnung von Kostenabrechnungen an die jeweiligen Benutzer

Die einfachste und gebräuchlichste Methode ist, die Eingabe eines persönlichen Schlüssels zu fordern. Nur wenn dieser Schlüssel (meist eine Kombination aus einem eindeutigen Benutzernamen und einem Passwort) mit einem der am Server gespeicherten Benutzer-Schlüssel übereinstimmt, wird der Zugriff auf die gemieteten Applikationen und Daten erlaubt. Zur Verbesserung der Sicherheit wird die Übertragung der Authentifizierungsdaten verschlüsselt. Zudem werden diese Daten am Clientgerät nicht dauerhaft und während der Applikationssitzung nach Möglichkeit nur codiert gespeichert. Zugriffsverwaltung macht also nur Sinn, wenn sie durch entsprechende Sicherheitsmaßnahmen (werden in Kapitel 2.6.5. behandelt) unterstützt wird.

Für ASP muss der Zugriff auf folgende Komponenten abgesichert werden:

- Netzwerk-Ressourcen (Kabel, Server-Netzwerk): Für die Vergabe von Berechtigungen ist der ISP bzw. NSP zuständig.
- Server-Ressourcen (Dateisysteme, Datenbanken, Prozessorleistung)
- Software-Ressourcen (Serverbetriebssystem, Applikationen)
- Personalisierte Daten (Name, Zertifikate, Applikationsdaten, "Eigene Dateien", Programmeinstellungen, System- und Accounteinstellungen, Protokolle, Kostenaufstellungen)
- Dienstleistungen (Datensicherung, (automatische) Hilfeleistungen und Beratung. z.B. per E-Mail)

Beim Vertragsabschluss mit dem ASP werden die Berechtigungen für die obigen Komponenten erteilt. Im Berechtigungsprofil eines Benutzers oder einer Benutzergruppe sind der jeweilige Zeitraum und das Ausmaß, in dem bestimmte Ressourcen verwendet werden dürfen, definiert. Ein solches Berechtigungsprofil muss sorgfältig mit den Lizenzvereinbarungen und den Zahlungsmodalitäten der gemieteten Applikationen abgestimmt sein. Das Betriebsystem bzw. die ASP Basistechnologie muss bei einem Zugriff auf eine Ressource durch einen Benutzer das jeweilige Berechtigungsprofil überprüfen und den Zugang entsprechend gewähren oder verwehren.

2.4. Plattformunabhängigkeit und Integration

Plattformunabhängigkeit (oder "Geräteunabhängigkeit") von ASP bedeutet die Betriebsmöglichkeit einer ASP Lösung auf unterschiedlichen Hard- und Softwareplattformen.

Integration einer ASP Lösung bezeichnet die Möglichkeit zur Zusammenarbeit mit anderer Software ("anders" im Sinne von anderen verwendeten Technologien und Schnittstellen).

Diese beiden Begriffe beschreiben ähnliche Problembereiche: Es geht darum, dass eine ASP Lösung zweckmäßigerweise unter möglichst vielen verschiedenen Hard- und Softwarekonfigurationen läuft.

2.4.1. Plattformunabhängigkeit

Das ASP-Konzept bietet a priori sehr gute Voraussetzungen für plattformunabhängiges Arbeiten aus Sicht der Anwender: Der zentrale Betrieb (Anwendungsfunktionalität und Bereitstellung) und das zentrale Management (Update, Programm- und Datenpflege) geschieht für alle Benutzer auf einer homogenen Serverplattform. Diese Applikationslogik am Server braucht deshalb selbst nicht unbedingt plattformunabhängig zu sein, da das Serverbetriebsystem oder die Serverplattform i.A. selten gewechselt wird. Für den Provider ist es aber bei einem Umstieg auf eine andere Serverplattform eine Erleichterung, wenn die verwendeten Technologien auch auf der neuen Plattform verfügbar sind und der Applikationsquellcode nicht oder kaum angepasst werden muss.

Besonders interessant erscheint Plattformunabhängigkeit aber auf Client-Seite. So kann ein großer Kundenkreis erschlossen werden und die Kunden können während des Softwaremietverhältnisses bedenkenlos auf andere Hardware oder Betriebsysteme umsteigen. Hinzukommen die nicht-technischen Ziele von Plattformunabhängigkeit (siehe Kapitel 1.3.1., "Geräte- und Ortsunabhängigkeit").

Grundsätzlich können natürlich nur Plattformen unterstützt werden, die mindestens einen Internetzugang mit ausreichend Bandbreite bieten.

Plattformunabhängigkeit kann prinzipiell auf zwei Arten erreicht werden: "Echte" Geräteunabhängigkeit liegt vor, wenn nur standardisierte Kommunikationsprotokolle, Schnittstellen und Sprachen verwendet werden. Dies ist bei ASP über den Internetbrowser (via Websprachen) der Fall. HTTP, HTML, Java (theoretisch plattformunabhängig, praktisch nein, z.B. wegen unterschiedlichen Dateisystemen [Paiha 2001]) und JavaScript (wenn man sich auf den ECMA Standard beschränkt) sind standardisierte Protokolle und Sprachen, die auf verschiedenen Internetbrowsern laufen, womit Plattformunabhängigkeit erreicht wird. Solcherart entworfene Applikationen sind jedoch ein für alle Mal für den Einsatz über einen Internetbrowser festgelegt, mehr dazu später.

Die zweite Methode zur Erreichung von Quasi-Plattformunabhängigkeit ist die, eine Technologie oder Software für möglichst viele verschiedene Systeme anzupassen und anzubieten. Die heutigen Internetbrowser sind ein gutes Beispiel: Für jede Hardwareplattform (PC (Intel), Apple, Handheld (RISC) etc.) und für jedes Betriebsystem (MS Windows, MacOS, Linux, Windows CE etc.) müssen eigene Browser zur Verfügung gestellt werden. Da es aber fast für alle Systeme Internetbrowser gibt, fällt dies nicht so sehr ins Gewicht.

Dieselbe Situation liegt vor, wenn der ASP oder der Hersteller der zugrundeliegenden ASP Basistechnologie nichtstandardisierte Kommunikationsprotokolle, Schnittstellen und Sprachen für die clientseitige Unterstützung einsetzt (wie es z.B. bei der Terminaltechnologie der Fall ist), aber für die gängigsten Systeme Kontrollprogramme und Browser-Plug-Ins anbietet. Plattformunabhängigkeit wird erreicht, obwohl die darunterliegenden Technologien nicht plattformunabhängig sind, weil sie entweder nicht standardisiert sind oder weil sie keine große Verbreitung oder Akzeptanz finden.

Folgende Tabellen geben Aufschluss über die Plattformunabhängigkeit von gebräuchlichen Client-Technologien und die Verbreitung im Zusammenhang mit ASP-Angeboten.

Die Client-Front-Ends sind zwar nicht standardisiert, sie bedienen sich aber teilweise standardisierter Technologien:

Name der Technologie oder des Programms	Menge der unterstützten Client-Plattformen	Menge der für ASP relevanten Client-Technologien	Verbreitung	Hersteller
AOL Netscape	Sehr groß	HTTP1.1, HTML, JavaScript (ECMA-Standard), Java, Plug-Ins	Sehr stark	AOL, Sun (Java)
MS Internet Explorer	Sehr groß	HTTP1.1, HTML, Jscript (ECMA-Standard), Java, ActiveX, VB Script, XML	Sehr stark	Microsoft, Sun
MS Windows Terminal Services	Sehr klein, nur MS Windows	RDP (Version 5.0); für browserbasierten Terminalclient: Plug-In, AciveX, Java	Stark	Nur Microsoft, Citrix
Citrix MetaFrame	Sehr groß	ICA; für browserbasierten Terminalclient: Plug-In, ActiveX, Java	Sehr stark	Citrix
SCO Tarantella	Sehr groß	AIP, Java	Gering	SCO

Tabelle 3 : Plattformunabhängigkeit von Client-Front-Ends

Standardisierte (oder de-facto standardisierte) Technologien:

Name der Technologie	Menge der unterstützenden ASP-Clientsysteme und deren Hersteller	Verbreitung
HTTP	Sehr groß	Stark
HTML	Groß, hauptsächlich Internetbrowser	Stark
JavaScript	Groß, hauptsächlich Internetbrowser	Stark
Java	Sehr groß	Gering
Microsoft RDP-Protokoll	Noch klein, Microsoft WTS und SCO Tarantella	Stark
Citrix ICA-Protokoll	Noch sehr klein, nur von Citrix	Sehr stark
SCO AIP-Protokoll	Noch sehr klein, nur von SCO Tarantella	Gering
WAP (Handy)	Klein, hauptsächlich Handys	Gering
CORBA	Sehr klein	Gering
SOAP	Sehr klein	Noch sehr gering
XML	Noch sehr klein (Internet Explorer)	Noch sehr gering

Tabelle 4 : Standardisierte Client-Technologien

Die Spalte "Verbreitung" bezieht sich hier auf den Anteil an den ASP-Angeboten, nicht auf die weltweite Verteilung der Art der Applikationsnutzung bei Unternehmen.

Internetbrowser, obwohl selbst nicht direkt plattformunabhängig, bieten durch ihre weite Verbreitung, die Verfügbarkeit für fast alle Systeme und vor allem durch den Einsatz von etablierten Schnittstellen und Standards wie HTTP und HTML, eine gute Basis für plattformunabhängiges ASP. Mit clientseitigem HTML alleine lassen sich jedoch nur sehr einfache Applikationen realisieren. Für komplexere Aufgaben kann man JavaScript und Java hinzunehmen, durch den Einsatz von Java kann man außerdem der Seitenorientierung (siehe Kapitel 2.1.2, "Websprachen") aus dem Wege gehen. Leider haben Java-Applets eine relativ schwache Performance und sind gemessen an der Funktionalität relativ groß. Es kommt zu

längeren Downloadzeiten, und das bei jedem Aufruf der Applikation. Aus diesen Gründen konnte sich Java als Clienttechnologie, sieht man über die Java-basierten Terminalclients hinweg, im ASP Bereich leider noch nicht durchsetzen.

In der Praxis zeigt sich, dass jeder Browser die Sprachen HTML und Javascript (geringfügig) unterschiedlich visualisiert und interpretiert. Es gibt sogar ganz gravierende Unterschiede zwischen IE und Netscape [Münz 2001][Paiha 2001] sodass, um die Lauffähigkeit in allen Browsern zu garantieren, die Clientlogik für jeden Browser separat programmiert werden muss, oder die Funktionalität extrem eingeschränkt werden muss. Es geht sogar so weit, dass sich gleiche Browser mit derselben Versionsnummer auf Apple's Mac und einem PC anders verhalten. Wie die Plattformunabhängigkeit von HTML, JavaScript und Java aus Anwendersicht zu bewerten ist, hängt letztendlich auch davon ab, wie der Programmierer mit den Browserunterschieden umgeht.

Mit Ausnahme von Java birgt der Einsatz von Websprachen außerdem, wie aus der Tabelle 4 ersichtlich, noch einen Nachteil: Eine in speziellen Websprachen implementierte Applikation ist für den Einsatz in heutigen Browsern entworfen. Sollen plötzlich andere Client-Websprachen oder -Laufzeitumgebungen und Übertragungsprotokolle unterstützt werden, so sind solche Umstellungen mit großem Aufwand und einem generellen Redesign der kompletten Applikation verbunden, wobei die Grenzen der Plattformunabhängigkeit mit Websprachen und Browsern ersichtlich werden.

Wenn aber bereits beim Applikationsdesign die Zusammenarbeit mit möglichst vielen Endgeräten und Browsern ins Auge gefasst wird, zeigt sich hier sogar ein Vorteil der Websprachen: Wenn implementiert, kann ein Webserver je nach Leistung des Endgerätes (und dessen Browser), falls die Client-Konfiguration dem Server bekannt ist, individuellen Code (z.B. in WAP (Wireless Application Protocol-Markupsprache für Handies)) übertragen. Bei den heutigen Handies und Palmtops mit ihren kleinen Displays und der relativ niedrigen Bandbreite ist das ein wichtiger Aspekt. Für mobile Geräte erscheinen Übertragungsprotokolle mit Semantik, wie es die websprachenbasierten Dienste darstellen, geeignet. Von Microsoft gibt es z.B. den "Mobile Information Server" [Microsoft 2002a]. Für dieses Serverprodukt können mithilfe von Visual Studio.NET (Kapitel 3.2.) und dem "Mobile Internet Toolkit" mobile Webapplikationen entworfen werden, die sich automatisch auf die Fähigkeiten des Endgerätes anpassen können. Die Benutzerschnittstelle wird dabei automatisch in dem für das Endgerät geeigneten Format (WHTML, HTML 3.2 oder "compact HTML") übertragen.

Abschließend lässt sich also über clientseitige Websprachen sagen, dass diese Technologien zwar theoretisch Plattformunabhängigkeit bieten, jedoch z.Z. nur von der Plattform "Internetbrowser" und von WAP-Browsern unterstützt werden und die Internetbrowser die Standards nicht vollständig implementieren. Die Plattformunabhängigkeit in der Praxis wird nochmals im einzelnen für die jeweiligen Technologien in Kapitel 3 diskutiert.

Auf der anderen Seite stehen wieder die **Systeme auf Terminal-Basis**: Auch hier wird u.a. der Internetbrowser als Plattform herangezogen. Es existieren i.A. Terminal-Clients für alle wichtigen Browser aber zusätzlich für zahlreiche darunterliegenden (Betriebsystem)plattformen. Wenn gar ein (wirklich geräteunabhängiger) Java-Terminalclient verfügbar ist, wie es bei Citrix MetaFrame und SCO Tarantella der Fall ist, lässt sich eine zukunftssichere Plattformunabhängigkeit erzielen. Bei Terminallösungen braucht nicht der Applikationscode geändert werden, wenn neue Clienttechnologien (auf Browser-Basis oder nicht) unterstützt werden sollen. Wenn das zugrundeliegende Kommunikationsprotokoll geräteunabhängig ist, wie es bei ICA von Citrix der Fall ist, braucht man nur ein entsprechendes neues Kontrollprogramm. Nicht immer ist jedoch jedem Benutzer die Installation von Software gestattet (die einmalige Installation des Client-Plug-Ins bzw. der ActiveX-Komponente muss unter Administratorrechten geschehen). Das Update der Clientlogik (z.B. auf eine höhere Version) funktioniert (auch aus letztgenanntem Grund) nicht bei allen Terminalsystemen automatisch im Hintergrund, bei den Websprachen jedoch bekommt man die Clientlogik bei

jeder Applikationssitzung immer in ihrer aktuellsten Form. Eine minimale clientseitige Wartung ist also bei den Terminals erforderlich. Bei der Auflistung der verwendeten Kommunikationstechnologien ist ersichtlich, dass es derzeit keine einheitlichen Standards gibt. ASP und Endkunde sind deshalb auf denselben Hersteller angewiesen. Die Versorgung mit clientseitigen Kontrollprogrammen ist aber zumindest bei Citrix und Tarantella ausgezeichnet. Bei Citrix werden für sehr viele Endgeräte Anpassungen zur Verfügung gestellt: Java, DOS, Win16/32, UNIX, Linus, Solaris [Citrix 2002]. Der Java-Client alleine bietet bereits Plattformunabhängigkeit, die systemspezifischen Terminaltreiber sind jedoch schneller. Bei SCO Tarantella gibt es nur einen Java-Client.

Die Kompatibilität zwischen Terminalclients und Browsern bzw. Betriebsystemen ist wesentlich besser als zwischen den Websprachen und den Browsern [Paiha 2001].

Die grafische Ausgabe bei Terminalprogrammen ist leider sehr starr, sie entspricht der Anzeige des Servers. Herkömmliche, für Desktop PCs entworfene Applikationen sind meist für Mindestauflösungen von 800x600 Bildpunkten und Eingabegeräte wie Maus und (große) Tastaturen konzipiert. Solche Applikationen lassen sich per Terminal auf Endgeräten geringer Leistung (wie z.B. Handys oder Palmtops) nicht komfortabel nutzen [Paiha 2001], da sich die Applikationen nicht an die Leistungen des Endgerätes anpassen können. Ein Terminal kann hier server- und clientseitig durch Reduzierung der Farben, Herunterskalieren der Bildschirmauflösung oder Scrolling nur unbefriedigend eingreifen. Für solche Fälle müssen eigene Applikationen geschaffen werden, was der Terminaltechnologie in Bezug auf Geräteunabhängigkeit einen Nachteil einräumt.

Abschließend lässt sich sagen, dass die Möglichkeit für plattformunabhängiges Arbeiten durch das ASP-Modell zwar gegeben ist, in der Praxis erscheinen aber weder reine Websprachen- noch Terminallösungen in ihrer derzeitigen Form dafür geeignet. Es gibt noch keine ausgereifte Technik, die es erlaubt, dieselbe Applikation, ohne dass diese im einzelnen speziell angepasst werden muss, gleichermaßen komfortabel, abwechselnd mit stationären und mobilen Endgeräten zu benutzen.

2.4.2. Integration

Integration ist unter den ASP Diensten ein bedeutendes Schlagwort. Es bedeutet die Zusammenführung von unterschiedlichen Softwarekomponenten und Technologien auf eine Plattform. In der ASP-Praxis verbergen sich dahinter drei verschiedene Bedeutungen:

1) Integration (Anpassung) der Applikation in (an) die ASP Basistechnologie

Wie bereits erwähnt, müssen Applikationen vor ihrem ASP-Einsatz auf "ASP Readyness" überprüft werden und gegebenenfalls angepasst werden. Zum Beispiel bietet IBM diesen Service als Teil ihres ASP-Enabling-Programm "ASP-Prime" an [DeGiglio 2000]. Eine Anpassung herkömmlicher Software an den webbasierten ASP Betrieb ist z.B. wegen der unterschiedlichen Entwicklungsplattformen, wegen der Seitenorientierung und wegen der Umstellungen bei der Anbindung an das Dateisystem sehr schwierig und wird deshalb kaum betrieben. Stattdessen werden ab und zu spezielle Versionen für den webbasierten Betrieb neu implementiert (z.B. die webbasierte Version von Microsoft Outlook). Die meisten webbasierten Anwendungen sind jedoch völlig neuartig und speziell für die Internetanwendung konzipiert.

Die Anpassung an eine Terminallösung ist hingegen relativ einfach zu bewerkstelligen und wird auch sehr oft betrieben.

2) Integration von mehreren Applikationen eines ASPs oder Integration mit Modulen, Tools, Utilities und Spezialhardware auf Server-Seite

Je größer und komplexer ein ASP-Softwarepaket, desto wichtiger ist serverseitige Integration. Besonders Software zur Automatisierung von Geschäftsabläufen (wichtigstes Beispiel: SAP) erfordert eine reibungslose Zusammenarbeit aller Programmmodule. Bei der Installation dieser Software, sei es lokal oder beim ASP müssen viele Applikationsmodule und Hardwarekomponenten eingesetzt werden, die nicht notwendigerweise von demselben Hersteller stammen. In diesem Zusammenhang wird auch das Wort "**Customization"** verwendet. Customization ist die Anpassung der Software an den Kunden. Wie bereits in der Einleitung erwähnt, ist diese bei ASP eher gering ausgeprägt, bei großen, unternehmenskritischen Applikationen aber hoch erwünscht. Die Prozedur der Customization geht zentralistisch viel schneller vonstatten als bei herkömmlicher Software, denn der Vorgang muss nur auf einem Server erledigt werden. Außerdem schafft sich der ASP die hard- und softwaretechnischen Voraussetzungen für reibungslose Installation und Betrieb selbst, und braucht auf die Hard- und Softwarekonfiguration des Kunden nur wenig Rücksicht nehmen. Dies spart für ASP und Kunde viel Zeit. Vor allem bei einer starken Filialstruktur des Kunden fällt dies ins Gewicht. Ansonsten unterscheidet sich die Kundenanpassung nicht von jener bei lokal installierter Software, wo diese vor Ort geschieht.

Ein weiterer Punkt von Integration betrifft den Einsatz zahlreicher unterstützender Server-Tools. Die Installation und Einrichtung dieser Software geschieht meist vor der Inbetriebnahme und vor dem Vertragsabschluß mit den Kunden, ist für diese unsichtbar und von ihnen unabhängig. An (mit der ASP Basistechnologie) zu integrierender Software wären als wichtigste zu nennen: Virenscanner, RAID-Treiber, automatische Backup Utilities, Tools zur Protokollierung und Messung von Ressourcenverbrauch (z.B. Performance, Netzwerkverkehr, Speicherbedarf), Software zur Speicherung und Verwaltung der Accounts (Directory-Services), Security-Tools (Verschlüsselung, Zugangsbeschränkung, Firewalls), Service Level Measurement und Operation Management.

Hinzukommt die Integration in die Hardwareplattform. Der Hersteller muss a priori natürlich eine Anpassung der ASP Basistechnologie für die gewünschte Hardwareplattform anbieten. Citrix bietet z.B. Spezialanleitungen für die Installation ihres Produktes "MetaFrame" auf konkreten Server-Modellen [Citrix 2002].

3) Integration von lokal installierter Software oder Schnittstellen zu Applikationen eines anderen ASPs.

Der Bedarf nach Integration von clientseitiger Software ins ASP-Gesamtsystem ist von Kundenseite her eher gering. Genau diesem Problem möchte man bei ASP ja aus dem Weg gehen. Wenn ein Kunde nach einer speziellen Zusatzfunktionalität verlangt, kann ihm diese, nach Prüfung der Sicherheitsanforderungen, vom ASP im Einzelfall zur Verfügung gestellt werden. Dieses Service geschieht auf Serverseite und wird meist gesondert in Rechnung gestellt.

Die Bindung an mehrere ASPs wird bei zunehmendem Angebot für die Kunden interessanter werden, die ASPs wollen jedoch mit Hilfe von starker Kundenbindung eine solche Vorgangsweise erschweren.

Zur Zeit sieht es leider sowohl bei websprachenbasierten- als auch bei terminalbasierten Diensten so aus, dass eine Integration von lokal installierter Software kaum stattfindet. Das bedeutet nicht, dass sie nicht möglich wäre. Im Terminalsystem von Citrix sind die meisten Hardwarekomponenten des Clientgerätes (Laufwerke (Magnetplatten, Diskettenlaufwerke, etc), Drucker, COM-Schnittstellen) integriert. Terminallösungen bieten wenigstens die

Möglichkeit, unternehmenskritische Daten bei Bedarf auf der lokalen Festplatte zu speichern. Außerdem wird Copy/Paste von und zum Clientgerät unterstützt. Im Protokoll von Citrix ist auch eine Möglichkeit vorgesehen, beliebige benutzerspezifische Daten zu übertragen und damit eine Anbindung an Clienthardware oder -software zu erreichen. Zum Beispiel könnten über eine solche Schnittstelle beliebige XML-Daten übertragen werden. Diese Technik der "Virtual Channels" wird, neben anderen Aspekten von Citrix, im praktischen Teil behandelt.

Das Wort "Integration" wird bei Websprachen eigentlich nicht gebraucht. Im Prinzip ist ein Zugriff auf lokale Ressourcen aber möglich: Ein Java-Applet kann auf diese zugreifen, sobald der Benutzer einmal seine Zustimmung dafür erteilt. Eine so komfortable Anbindung an Clienthardware wie bei den Terminals gibt es aber (noch) nicht. Die Kooperation (d.h. der Datenaustausch) zwischen webbasierten Diensten von verschiedenen ASPs funktioniert genauso schlecht wie bei den terminalbasierten Angeboten. Bei der Zusammenführung von ASP-Angeboten ist man auf Kooperationen von Seite der ASPs angewiesen. Sind solche nicht vorhanden, ist man mit Softwarekomplettlösungen aus einer Hand besser bedient als zu versuchen, einzelne Module verschiedener Anbieter zusammenführen zu wollen.

2.5. Verrechnungstechnologien

Als die ASP-Idee geboren war, wurden rasch verschiedene theoretische Abrechnungsmodelle vorgeschlagen. Man sprach von nutzungsabhängigen Gebühren, nach Zeit oder Anzahl von (nicht näher definierten) "Transaktionen" in Kombination mit einer fixen monatlichen Grundgebühr. Die anfängliche Definition einer Transaktion in Form von Mausklicks und Tastaturanschlägen erwies sich jedoch als unzureichend. Weder für ASPs noch für Kunden schien dies ein geeignetes Abrechnungsmodell zu sein.

Aus diesem Grund sind z.Z. die monatlichen Flatrates am meisten verbreitet. Sie sind sowohl in organisatorischer als auch in technischer Hinsicht leichter zu implementieren und erlauben eine einfache Kalkulierbarkeit der Kosten für den Kunden. Oftmals werden einfach herkömmliche Abrechnungssysteme aus dem Telekommunikationsbereich eingesetzt.

Auf lange Sicht gesehen, um auf diesem immer mehr vom Konkurrenzkampf dominierten Markt zu reüssieren, braucht ein ASP jedoch ein eigenes ASP-Abrechnungssystem, das den Anforderungen verschiedenartiger Preisgestaltung, Märkte, Applikationsangebote und Dienstleistungen gewachsen ist [ModusNovo 2002a][ModusNovo 2002b]. Bei der Angebotsstellung und Preisgestaltung wird eine ähnliche Entwicklung wie bei der Deregulierung des Telefoniesektors erwartet, mit immer differenzierteren Angeboten und komplexeren Preisschemas.

ASP-Abrechnungsysteme müssen detaillierte Benutzungsinformationen sammeln können, halbautomatisch attraktive Preisschemas generieren und entsprechende Rechnungen ausstellen können.

Die Entscheidung, welche Kosten von allen Benutzern getragen werden (durch eine fixe monatliche Grundgebühr) und welche Dienste individuell verrechnet werden (durch nutzungsbasierte Gebühren) muss unter Gesichtspunkten ausgefeilter Marktstrategien geschehen.

ASP-Abrechnungssysteme unterscheiden typischerweise zwischen drei verschiedenen Arten von Gebühren. Jede dieser Komponenten kann, je nach Angebot, auch optional sein:

- **Einmalige Gebühren**

  Darunter fallen z.B. eine Registrierungsgebühr für einen individuellen Benutzer oder eine Einrichtungsgebühr für die Servereinrichtung für einen großen Kunden.

Manchmal, wenn der ASP zusätzlich Hardware mitvermietet, fallen darunter auch Kautionen für die Client-Hardware.

Kosten für Erstinstallation, Integration und Einschulungen werden hingegen meist nicht von den einmaligen Gebühren getragen.

- **Wiederkehrende Gebühren**

  Sie werden regelmäßig eingehoben und sind von der tatsächlichen Nutzung unabhängig. Ein typisches Beispiel für wiederkehrende Gebühren ist eine monatliche Zugangsgebühr für eine Applikation oder eine Leistung.

- **Nutzungsabhängige Gebühren**

  Nutzungsgebühren werden normalerweise in nahezu Echtzeit generiert und assoziieren Ereignisse mit Nutzung. In der ASP-Umgebung wäre eine Gebühr pro Zugriff auf eine Applikation eine nutzungsbasierte Gebühr.

2.5.1. Verrechenbare Ereignisse in einer ASP-Umgebung

Es gibt eine ganze Reihe von Tätigkeiten, Vorgängen und Ereignissen, die ein ASP an seine Kunden weiterverrechnen kann und muss. Je höher die Granularität, um so besser können die Angebote auf die Kunden zugeschnitten werden.

- **Allgemeiner ASP Betrieb**

  Hier zu nennen wären u.a. Kosten für Material und Personal sowie Abschreibungen, Zinsen, Steuern und sonstige betriebliche Aufwendungen, wie z.B. Miete, Versicherungen, Subhonorare. Diese Kosten fließen normalerweise nicht in die nutzungsabhängige Gebühr ein.

- **Weiterverrechnung von Diensten**

  Hier fließen die Zahlungen ein, die ein ASP an seine Partner (NSPs, ISVs, Kreditkartenunternehmen, u.v.a.m.) abliefern muss. Auch diese Kosten werden zum Großteil von den wiederkehrenden Gebühren getragen.

- **ASP Dienstleistungen**

  Hier zu nennen wären Schulungen, Softwarewartung und Kundenservice, kurz, alles was ASP-Personal für die Kunden leistet. Hier gibt es eine Reihe von Dienstleistungen, die explizit in Form einer Aufzahlung an die Kunden weiterverrechnet werden ("Extended-Services").

- **Automatisiertes Kundenservice**

  Ein Kunde kann über eine beliebige Schnittstelle (Briefverkehr, Telefon, Internet, etc.) ein verrechenbares Ereignis auslösen, z.B. Hinzufügen eines Users, eine Zugangskonfiguration zu einer Applikationen ändern oder eine Rechnung drucken.

- **Infrastruktur**

  Die Gebühren entstehen hier durch die Netznutzung und Servernutzung sowie deren Infrastruktur. Das Abrechnungssystem kann CPU-Nutzung, Zugangszeit, Datenbanktransaktionen, Datenträger- oder Netzwerknutzung aufnehmen, analysieren und verrechnen.

- **Innerhalb der Applikationen auftretende Ereignisse**

  Öffnen oder Anlegen einer Datei, Aufruf der Hilfefunktion oder eines Utilities sowie das Hinzufügen eines Eintrages in einer Datenbank sind Beispiele für applikationsbezogene Ereignisse.

2.5.2. Komponenten eines ASP-Abrechnungssystems

Anhand von Abbildung 2-4 sollen kurz die Komponenten eines ASP-Billingsystems beschrieben werden und wie diese untereinander, mit dem Internet und der gehosteten Applikation kommunizieren:

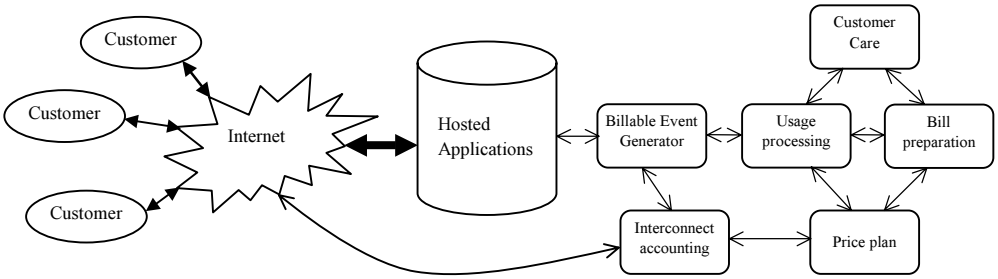

Abbildung 2-4 : Komponenten eines ASP-Abrechnungsystems [ModusNovo 2002a]

- **Billable event generator**

  Er sammelt Protokolldaten technischer Art (aus Dateien oder Datenbanken), die ihm Soft- und Hardwareressourcen (vor allem die Service Level-, Operation Level-, und Subscriber Management Software) zur Verfügung stellen und identifiziert die Nutzung dieser Ressourcen auf Benutzerebene. Hierzu fasst er mehrere technische Ereignisse zusammen und rekonstruiert daraus die getätigten Benutzervorgänge.

  Diese Komponente erkennt auch die Aktivierung von Funktionen innerhalb von Applikationen, was von immenser Wichtigkeit ist.

  Sie ist die intelligenteste und technologisch herausforderndste Komponente des Abrechnungssystems.

- **Usage processing**

  Diese Einheit empfängt die Nutzungsdaten von dem Ereignis-Generator und verarbeitet sie unter Zuhilfenahme des für den jeweiligen Kunden definierten Algorithmus. Dies geschieht unter Mithilfe der Preisplan-Applikation.

- **Price plan**

  Der Preisplaner ermöglicht den Marketing-Spezialisten des ASPs Preisgestaltungen für bestimmte Märktsegmente zu definieren und jede Art von ASP-Kosten in einen Abrechnungsalgorithmus einfliessen zu lassen. Hier können nun auch Kosten des allgemeinen ASP Betriebes, die Weiterverrechnung von Diensten und die ASP Dienstleistungen (siehe Kapitel 2.6.4.) halbautomatisch erfasst werden.

- **Bill Preparation**

  Die gewichteten verrechenbaren Ereignisse werden für einen bestimmten Rechnungszeitraum zusammengestellt. Danach erzeugt diese Einheit eine Rechnung für den Kunden. Dabei können auch verschiedene Ermäßigungen, die nur am Ende eines Rechnungszeitraumes angewendet werden können, berücksichtigt werden. Der Rechnungsgenerator formatiert die Rechnung, sodass sie für den Kunden sinnvoll ist.

- **Customer care**

  Das Kundenservice-System ist die interaktive Schnittstelle zum Kunden, im Prinzip ein Teil des ASP-Dienstes. Mittels dieser Applikation (meist webbasiert) hat der Kunde jederzeit Einblick in Kostenaufstellungen, Rabatte, Guthaben und Rückstände. Diese Einheit kann auch in die Subscriber- und Service Level- Management Software integriert sein.

- **Interconnect accounting**

  Diese Komponente stellt webbasierte Dienste anderer Service Provider in Rechnung.

2.5.3. Verrechnung von Applikationsnutzung

Ein Teil des "Billable event generators", nämlich jener, der Ereignisse innerhalb Applikationen sammelt, soll genauer untersucht werden. Er wird als einziger weder von externer Software noch direkt von der Applikation mittels Log-Daten versorgt und stellt aus diesem Grund eine der größten Herausforderungen an ein ASP-spezifisches Billingsystem dar. Die ASP-Anwendungen protokollieren i.A. nicht detailliert ihre Nutzung.

Die Herausforderungen an den Ereignisgenerator sind folgende:

- Der Rekorder muss generisch sein, d.h. er muss die Nutzung der Applikation identifizieren können, ohne die Funktionalität der Applikation kennen zu müssen. Er muss für jede Art von Applikation (terminal- oder webbasiert) universal sein und muss sich ohne Änderung von sich selbst oder der Applikation integrieren lassen.

- Er darf die Performance der Applikation nicht beeinflussen.

- Er muss Ereignisse, die nicht verrechnet werden, ignorieren. Zum Beispiel sollte der Rekorder das "mouse-move"-Ereignis der GUI ignorieren.

- Schlussendlich-und am komplexesten-muss es einen Mechanismus geben, der die rohen, technischen Applikationsereignisse, wie Systemprozeduraufrufe oder Ein-/Ausgabe, in Semantik umsetzt, die die Benutzervorgänge reflektiert. Dazu ist ein Lernmodus vorgesehen, in dem der Ereignisgenerator bei Auslösen einer Applikationsfunktion die auftretenden technischen Ereignisse zusammenfasst. Sie werden anschließend von Hand mit einem Benutzervorgang assoziiert.

Während des Betriebs werden dann die Benutzervorgänge aus den technischen Ereignissen mittels Mustererkennung, statistischen Verfahren und neuronalen Netzen identifiziert.

2.6. Server Herausforderungen

2.6.1. Server Architektur und Applikationsprogrammlogik

2.6.1.1. Server Architektur

Alleine durch die geographische Trennung von Applikation (beim ASP) und Benutzerschnittstelle (beim Anwender) sind Grundzüge der ASP-Gesamtarchitektur bereits vorgegeben.

Eine Einteilung in Schichten kann nach logischen (Daten- und Applikationslogik, Schnittstellen) bzw. physikalischen Gesichtspunkten (physikalisch getrennte Geräte) erfolgen, ist jedoch nicht immer klar nachvollziehbar.

Grob gesprochen gibt es ein Endgerät (**Client,** auch **"Front-End"** genannt) mit Ein- und Ausgabeperipherie (Bildschirm, Touchscreen, Tastatur, Stift etc.), eine zentrale Rechnereinheit auf der die mehrbenutzerfähigen Applikationen laufen und die Daten bereitgehalten werden (**Server**) und einen Datenkanal (**Netz**), über den die Kommunikation abgewickelt wird. Unabhängig davon, aus wievielen Komponenten eine ASP-Architektur im konkreten technischen Fall aufgebaut ist, lässt sich das ASP-Modell als **Client-/Servermodell** bezeichnen.

Betrachtet man die Serverlogik alleine, so lassen sich dort weitere grobe Unterteilungen vornehmen:

Im einem traditionellen websprachenbasierten System oder terminalbasierten System befinden sich hierbei jeweils der Web Server oder der Terminalserver ("**Middle-Tier**" oder schlicht "**Server**") sowie die Datenhaltung **("Back-End**": Datenbankserver, Dateisysteme oder ERP-Systeme) in einer physikalischen oder logischen Einheit. Der Web Server tritt hierbei als Vermittler zwischen Front-End, Applikationsprogrammlogik und Back-End auf. Das bedeutet, dass es i.A. keine direkte Anbindung des Clients an die Datenbank bzw. an das serverseitige Dateisystem gibt. Für die Kommunikation zwischen den Schichten dienen wohldefinierte Protokolle.

Man bezeichnet ein solches Modell als **3-Schichtenmodell** (Front-End, Middle-Tier, Back-End):

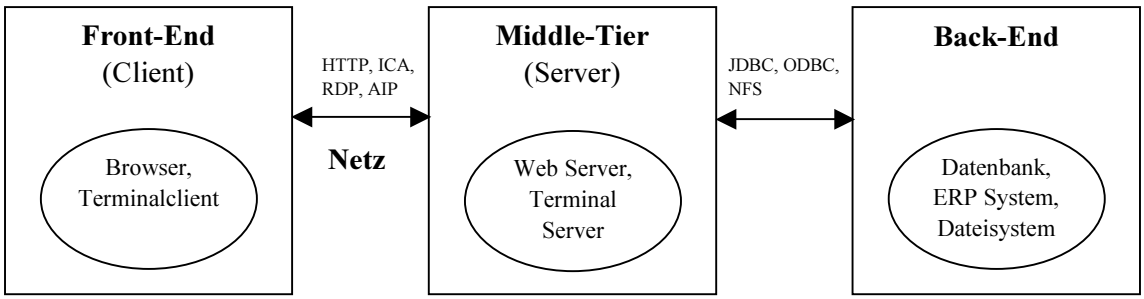

Abbildung 2-5 : 3-Schichtenmodell

Seit etwa 1998 gibt es für websprachenbasierte Systeme eine neue Architektur, die eine zusätzliche Soft- und Hardwarewarekomponente, den "Application Server" (allg. auch "**Middleware**" genannt) vorsieht und durch eine **verteilte**, **komponentenorientierte Anwendungsarchitektur** zahlreiche Vorteile bietet. Es gibt Application Server, die zusätzlich

zum Web Server betrieben werden (siehe Abbildung 2-6), oder die den Web Server integriert haben. Die Clientlogik wird hierbei zunehmend als "Black-Box" angesehen. Der Web Server hält gemischte client- und serverseitige Benutzerinterfaceprogrammlogik (siehe Kapitel 2.7.1.) bereit, die vor der Auslieferung an den Client mittels serverseitiger HTML-Codegenerierung in rein clientseitige Benutzerinterfaceprogrammlogik umgewandelt wird. Der Application Server hält nur Applikationsprogrammlogik (Softwaremodule für Berechnungen, Transaktionslogik, etc.) bereit. Im Back-End-Bereich, (Datenbankserver oder ERP-System, in der Grafik mit "Database Server" bezeichnet) werden persistente Daten bereitgehalten.

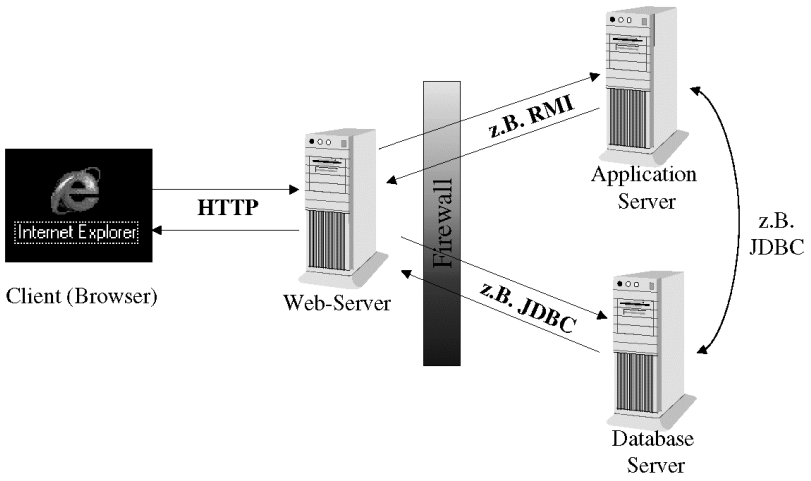

Abbildung 2-6 : Client-/Serverarchitektur mit einem Application Server

Diese vierschichtige Architektur bietet eine saubere Trennung von Benutzerinterfacelogik und Applikationslogik, bessere Skalierbarkeit durch Load-Balancing und bessere Wartbarkeit.

Zusätzlich integriert der Application Server low-level Funktionen wie Transaktionsmanagement, Sessionverwaltung, Multithreading, Ressourcenmanagement, Sicherheit (Authentifizierung, Autorisierung, Verschlüsselung), Server Management und -Monitoring und stellt sie als high-level Dienste zur Verfügung. Solche Dienste brauchen also nicht mehr für jede einzelne Applikation implementiert zu werden und können zur Laufzeit konfiguriert werden.

Im praktischen Teil wird die "ASP-Tauglichkeit" der herkömmlichen Web Server-Architektur ("Traditionelle Internettechnologien und -sprachen") mit drei Application-Server Produkten verglichen ("BEA WebLogic", "IBM WebSphere", "Microsoft .NET").

2.6.1.2. Applikationsprogrammlogik

Eng mit der Server Architektur ist die Applikationsprogrammlogik verbunden.

Bei terminalbasierten Systemen bspw., wo herkömmliche Windows- oder UNIX-Anwendungen über das Internet bereitgestellt werden, gibt es als einzige Vorgabe an die Applikationslogik, dass die jeweilige Applikation als ausführbare Datei realisiert sein muss.

Bei den websprachenbasierten Systemen hingegen definiert die Server Architektur gemeinsam mit den verwendeten Internetsprachen genau, wie eine websprachenbasierte Applikation aufgebaut sein muss oder soll. Die Applikationsprogrammlogik kann entweder gemeinsam in die serverseitige, HTML-generierende, seitenorientierte Benutzerinterfaceprogrammlogik integriert werden (etablierte Technologien hierzu sind CGI, PHP, ASP, JSP und Servlets) oder mithilfe von Modulen und Makros (JavaBeans, SSI) oder Komponententechnologien (EJB, COM, COM+) in eigene Dateien und Objekte ausgelagert werden. Komponentenmodelle erlauben eine bessere Modularisierung der

Applikationsprogrammlogik und bieten leistungsfähige Verwaltungsfunktionen. Möglich ist auch, dass zwar Benutzerinterfaceprogrammlogik und Applikationsprogrammlogik getrennt sind, diese sich jedoch optional in derselben Datei befinden können und durch spezielle Tags eingeleitet werden (Beispiel: ASP.NET).

Skriptsprachen wie SSI, ASP, JSP und PHP werden in HTML-Code an den gewünschten Stellen eingebettet, serverseitig ausgeführt und generieren entsprechenden HTML-Code. Programmiersprachen und -mechanismen wie CGI und Servlets hingegen stellen eine HTML-Seite über den Standardausgabekanal mittels entsprechender "Print"-Anweisungen zusammen. (siehe dazu ausführlich Kapitel 3.1.3.) Sie können die Anwendungslogik wesentlich einfacher integrieren.

Je nach Internetsprache ruft entweder die Benutzerinterfaceprogrammlogik Teile der Applikationsprogrammlogik auf (datenzentriert) oder umgekehrt (applikationszentriert). Bei den meisten Skriptsprachen wie SSI, ASP, JSP und PHP hat die Benutzerinterfaceprogrammlogik die Kontrolle, bei CGI, Servlets und ASP.NET liegt die Kontrolle bei einer ereignisgesteuerten (z.B. "Page_Load"-Ereignis) Applikationslogik, was viel sinnvoller und praktischer erscheint, da man dadurch nicht mehr an die sequentielle Abarbeitungsreihenfolge des HTML-Layouts gebunden ist.

Zum Vergleich ein einfaches Code-Beispiel in PHP und ASP.NET (Leider kann die Applikationsprogrammlogik nicht am gleichen Beispiel wie im Kapitel "Benutzerinterfaceprogramlogik" gezeigt werden, da die dort gezeigte ASP-Applikation nicht Open-Source ist.):

Es soll eine Zahl eingelesen werden und ihre Quadratwurzel ausgegeben werden. Die Benutzerschnittstelle soll eine Eingabezeile und eine Schaltfläche zur Einleitung der Berechnung haben.

- **Beispiel: PHP**

  Im Fall von PHP benötigt man eine HTML-Datei, die die Benutzerinterfaceprogrammlogik für die Eingabe enthält und eine PHP-Seite, die die Applikationsprogrammlogik und die Benutzerinterfaceprogrammlogik für die Ausgabe vermischt enthält.

  Code 1: ***QuadratwurzelEingabe.html*** :

```
<html>
<head><title>Quadratwurzel</title></head>
<body>
Quadratwurzel berechnen
<form method=post action="Quadratwurzel.php3">
<input type="text" name="op">
<input type="submit">
</form>
</body>
</html>
```

  Code 2: ***Quadratwurzel.php3*** :

```
<html>
<body>
<?php echo "Die Quadratwurzel aus $op ist:">
<?php echo sqrt($op)>
</body>
</html>
```

- **Beispiel: ASP.NET**

  Bei ASP.NET ist die Benutzerinterfaceprogrammlogik der gesamten Applikation in einer Datei (Quadratwurzel.aspx) und komplett von der Applikationsprogrammlogik (Quadratwurzel.aspx.vb) getrennt.

  Benutzerinterfaceprogrammlogik in ASP.NET:

  Code 3*: *Quadratwurzel.asmx:

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="Quadratwurzel.aspx.vb"
Inherits="ASPQuadratwurzel.Quadratwurzel"%>
<HTML>
<HEAD><TITLE>Quadratwurzel</TITLE></HEAD>
  <form id=WebForm1 method=post runat="server">
    <asp:label id=lblHinweis runat="server">Quadratwurzel
berechnen</asp:label>
    <asp:textbox id=txtEingabe runat="server"></asp:textbox>
    <asp:button id=btnSubmit runat="server" Text="Submit"></asp:button>
<P></P>
    <asp:label id=lblErgebnis runat="server"></asp:label>
  </form>
</HTML>
```

  Applikationsprogrammlogik in Visual Basic:

  Code 4: *Quadratwurzel.aspx.vb*:

```
Imports System
Imports System.ComponentModel.Design
Imports System.Data
Imports System.Drawing
Imports System.Web
Imports System.Xml
Imports System.Web.SessionState
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls
Imports Microsoft.VisualBasic

Public Class Quadratwurzel Inherits System.Web.UI.Page
    Protected WithEvents lblHinweis As System.Web.UI.WebControls.Label
    Protected WithEvents btnSubmit As System.Web.UI.WebControls.Button
    Protected WithEvents txtEingabe As System.Web.UI.WebControls.TextBox
    Protected WithEvents lblErgebnis As System.Web.UI.WebControls.Label
    Dim num As Double

#Region " Web Form Designer Generated Code "

    'This call is required by the Web Form Designer.
    <System.Diagnostics.DebuggerStepThrough()> Private Sub
InitializeComponent()
    End Sub

    Private Sub Page_Init(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Init
        'CODEGEN: This method call is required by the Web Form Designer
        'Do not modify it using the code editor.
```

```
        InitializeComponent()
    End Sub

#End Region

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        If Not IsPostBack Then   ' Evals true first time browser hits the
page
            lblErgebnis.Visible = False
        End If
    End Sub

    Private Sub btnSubmit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSubmit.Click
        txtEingabe.Visible = False
        lblHinweis.Visible = False
        btnSubmit.Visible = False
        num = Math.Sqrt(Double.Parse(txtEingabe.Text))
        lblErgebnis.Text = "Die Quadratwurzel aus " + txtEingabe.Text + "
ist: " + num.ToString()
        lblErgebnis.Visible = True
    End Sub

End Class
```

Das Ausmaß an Code für die webbasierte Berechnung der Quadratwurzel bei ASP.NET erscheint enorm, das meiste an Code stellt jedoch nur das automatisch generierte Grundgerüst dar, das bei größeren Anwendungen vernachlässigbar klein wird.

In Quadratwurzel.aspx.vb sind entsprechende Verweise auf die GUI-Elemente angelegt, um auf die Ereignisse und Attribute dieser Objekte Zugriff nehmen zu können.

Wie anhand der Methode "btnSubmit_Click" gut zu sehen ist, die beim Drücken des Buttons aktiviert wird, hat die Applikationsprogrammlogik die volle und einzige Kontrolle über Elemente der Benutzerschnittstelle.

Mit ASP.NET geschieht die Trennung von Benutzerschnittstelle und Applikationslogik am besten und saubersten.

2.6.2. Verfügbarkeit

Hochverfügbarkeit ist eine der größten Herausforderungen im technischen ASP-Bereich. Herkömmliche Internetangebote (vornehmlich Content ohne Qualitätsgarantien) stellten nur mittelmäßige Verfügbarkeitsanforderungen an die Provider. Umgekehrt waren die Ansprüche der Kunden an die Verfügbarkeit solcher Dienste meist nicht sehr hoch, da das Internet noch nicht zentraler Bestandteil der Geschäftsprozesse ("E-Business") und Applikationen geworden war.

Es ist jedoch leicht zu erklären, warum Hochverfügbarkeit ein wichtiges Thema bei ASP darstellt:

- Der Kunde erwartet sich vom ASP die Bereitstellung von Diensten, die garantierte Ergebnisse liefern. Aus diesem Grund gibt es die SLAs, auf deren Erfüllung die Kunden bestehen. Hochverfügbarkeit befähigt einen ASP, die vereinbarten SLAs einhalten zu können.

- Oft werden unternehmenskritische Daten und/oder Applikationen beim ASP (oder AIP) gehostet, auf die der Kunde nicht in der Lage ist zu verzichten. Bei einem ungeplanten

Ausfall hat der Kunde mittels seiner eigenen IT-Spezialisten keinen Einfluss auf die Wiederaufnahme seiner Applikationssitzung.

- Die Serverarchitektur für ASP ist wesentlich komplexer als jene für Content-Providing. Es gibt wesentlich mehr ausfallsträchtige Komponenten.
- Die Anforderungen an Applikationsverfügbarkeit sind beträchtlich höher als bei Datenverfügbarkeit. (Es gibt keine fehlerfreien Applikationen !)
- Die Serverressourcen werden von mehreren Benutzern gleichzeitig in Anspruch genommen. Restriktionen bei Security, Skalierbarkeit und Performance können die Vorhersagbarkeit und Verfügbarkeit des Systems negativ beeinträchtigen.
- Aus eben genannten Gründen ist das Risiko eines Fehlers (und damit möglicherweise eines Ausfalles) einer Komponente höher als bei Stand-Alone-Software. Der ASP muss mittels geeigneter fehlertoleranter und redundanter Komponenten diesem Problem gegensteuern, um trotzdem eine hohe "End-to-End"-Verfügbarkeit garantieren zu können. Idealerweise sollte der Benutzer von dem Ausfall einer Server- oder Netzkomponente nichts bemerken.

Bevor ins Detail gegangen wird, einige Definitionen [Kopetz 1996]:
Gegeben sei ein fixer Beobachtungszeitraum. Dann ist die *Verfügbarkeit* jener summierter Zeitanteil (in %), in dem das System in der Lage ist, seine Aufgabe zu erfüllen. Als *Up-Time* wird der entsprechende summierte absolute Zeitraum bezeichnet, als *Down-Time* die Gesamtzeit, in der das System nicht in der Lage ist, seine Aufgabe zu erfüllen.

Die *Verlässlichkeit* eines Systems ist hingegen die Wahrscheinlichkeit, dass ein System innerhalb des Beobachtungszeitraumes kontinuierlich in der Lage ist, seine Aufgabe zu erfüllen.

Ein unbeabsichtigter Zustand wird als *Fehler* (Error) bezeichnet. Die *Fehlerrate* berechnet sich aus der Anzahl der Fehler des Systems pro Zeiteinheit.

Die durchschnittliche Zeit bis zu einem Fehler wird als MTTF (Mean Time To Fail) bezeichnet, die *Wartbarkeit* ergibt sich aus der Durchschnittszeit, die für eine Reparatur, hervorgerufen durch einen Fehler, aufgewendet wird und wird als MTTR (Mean Time To Repair) bezeichnet.
Damit lässt sich *Verfügbarkeit* auch durch MTTF/(MTTF+MTTR) (also der Anteil der MTTF an der Gesamtzeit, bis zu einem Fehler) ausdrücken.
Im ASP-Bereich ist Verfügbarkeit vor allem durch hohe MTTF und eine hohe Verlässlichkeit (d.h. geringe Fehlerrate) anzustreben.

Eine geringere MTTR bei gleichzeitiger reduzierter MTTF würde zwar den gleichen Wert an Verfügbarkeit liefern, die Ausfälle würden sich jedoch häufen. Üblich sind End-To-End-Verfügbarkeiten zwischen 95% und 99,95%.

2.6.2.1. Ursachen für Stillstandszeiten

Downtime (Stillstandszeit) lässt sich in zwei Kategorien einteilen: **geplante** und **ungeplante Stillstandszeit** [ECO 2000]. Geplante Stillstandszeiten sind in den SLAs definiert und beeinträchtigen deshalb nicht deren Einhaltung. Ungeplante Stillstandszeiten können zu einer Verletzung der SLAs führen. Die Vermeidung von beiden Varianten von Stillstandszeiten hilft sowohl auf ASP- als auch auf Kundenseite (bei geschäftskritischen Anwendungen) die Geschäftseffizienz zu verbessern.

Geplanter Stillstand ist während der normalen Wartungsarbeiten der Applikation oder des Systems vorgesehen. Obwohl sie typischerweise zu Zeiten niedrigstmöglichem Applikationsbedarf getätigt werden, machen sie nahezu 80% der Gesamtstillstandszeit aus.
Das Spektrum an geplanter Downtime erstreckt sich über:

- Routinearbeiten für Datenbanken und Applikationen
- Periodische Wartung der Datenbanken, der Applikationen und des Netzwerks
- Neue Entwicklungen (Updates) bei Hardware, Betriebsysteme, Datenbanken, Applikationen und Netzwerk

Abbildung 2-7 [ECO 2000] zeigt die vielen Möglichkeiten von geplantem Stillstand:

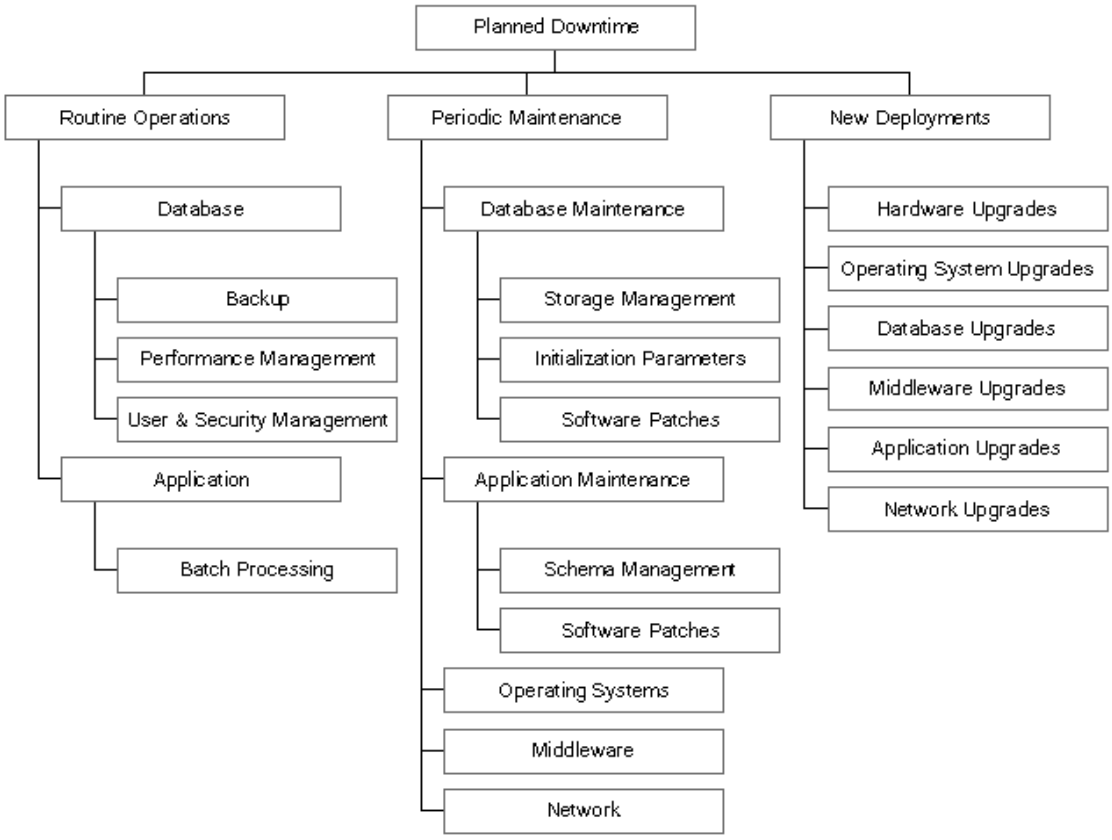

Abbildung 2-7 : Ursachen geplanter Stillstandszeiten [ECO 2000]

Die effizienteste Art, geplante Stillstandszeiten zu vermeiden ist, *kontinuierliche* Betriebsmethoden einzusetzen.

Ungeplante Stillstandszeiten werden durch eine Reihe von Ursachen hervorgerufen, im Groben sind dies:

- Betriebsystem-, Datenbank-, Applikations- und Netzwerkausfälle
- Diverse Hardwareausfälle
- Ausfälle hervorgerufen durch fehlerhafte oder unerlaubte menschliche Eingriffe
- Ausfälle hervorgerufen durch Katastrophen

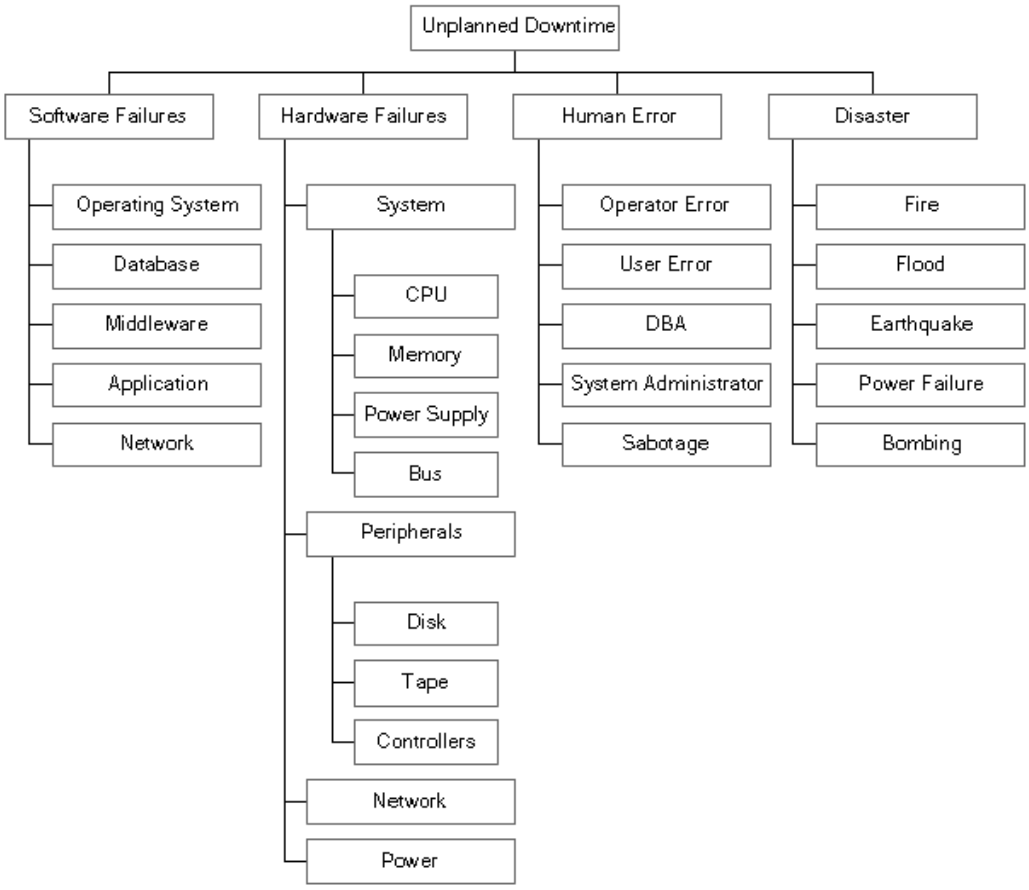

Abbildung 2-8 : Ursachen ungeplanter Stillstandszeiten [ECO 2000]

2.6.2.2. Maßnahmen gegen ungeplanten Stillstand

Prinzipiell gibt es zwei Möglichkeiten, die Verfügbarkeit zu erhöhen:

- **Verlängerung der Up-Time (Betrieb sicherstellen)**

  Dies wird erreicht, indem das Risiko von Ausfällen reduziert wird. Dabei helfen Mechanismen wie **Datensicherung,**. **Ausfallssicherheit** und **Fehlertoleranz.**

- **Reduzierung der Down-Time (Betriebsunterbrechungsphase verkürzen)**

  Diese kann verkürzt werden, indem rasche und effektive Maßnahmen zur **Wiederherstellung** (Recovery) eingesetzt werden (Backups, Hot Stand-By Komponenten).
  Läuft die Wiederherstellung parallel zu den Ausfallssicherungs- und Fehlertoleranzmechanismen, wird die Stillstandszeit zusätzlich verkürzt.

Idealerweise ergänzen sich beide Maßnahmen. Die einzelnen Prozesse werden nun genauer vorgestellt.

- **Datensicherung**

  Ein robuster ASP Betrieb beginnt mit einer effektiven Sicherung von Applikationen und Daten vor potentiellem Datenverlust. Abzuschätzen sind sowohl die Wahrscheinlichkeit eines Datenverlustes als auch die dabei auftretende Menge an Datenverlust.

Die wahrscheinlichsten Ursachen von Datenverlust sind (siehe "Ursachen ungeplanter Stillstandszeiten"):

- Komponentenfehler (Hardware, Software)
- Fehlerhafte oder unerlaubte menschliche Eingriffe
- Katastrophen

Es ist einsichtig, dass es bei Datenverlust zu Ausfällen kommen kann, der deshalb durch geeignete Sicherungsmechanismen verhindert werden sollte. Diese werden ausführlich im Kapitel "Sicherheit" behandelt.

Aber auch das Absichern entsprechender Daten-Redundanz (Backups, Mirroring), die sich die Ausfallssicherheits- und Fehlertoleranzmechanismen zunutze machen, ist Aufgabe der Datensicherung.

- **Ausfallssicherheit und Fehlertoleranz**

Load-Balancing verteilt die Last (Netz-, Server-, Prozessorlast) gleichmäßig auf mehrere Komponenten und verhindert dadurch Ausfälle durch Überlastung.

Fehlertolerante Systeme bringen sich automatisch nach einem Soft- oder Hardwarefehler wieder in einen konsistenten Zustand (z.B. automatischer Neustart). Durch **Redundanz** (Mirroring) werden Soft- und Hardwarekomponenten sowie Daten doppelt oder mehrfach abgebildet, vor dem Benutzer jedoch als ein **virtuelles System** dargestellt. Im Fehlerfall übernimmt eine andere Komponente die Arbeit, während an der Reparatur der Primärkomponente gearbeitet wird.

Durch **Clustering** lassen sich verschiedene Rechner zu einer logischen Einheit zusammenfassen. Vor allem bei gemeinsamen Ressourcen wie z.B. Dateiservern (bei Terminaldiensten) und Datenbankservern (bei webbasierten Diensten) wird dies oft betrieben, da durch die Verteilung die Wahrscheinlichkeit eines "Single-point-of-failures" verringert wird.

Die größte Herausforderung besteht darin, dass ein Fehlerfall oder Ausfall einer Soft- oder Hardwarekomponente aus Benutzersicht nicht bemerkt wird, und somit die vereinbarte End-to-End Verfügbarkeit nicht negativ beeinflusst wird.

- **Wiederherstellung**

Ist es bereits zu einem Ausfall gekommen, muss danach getrachtet werden, die Down-Time möglichst kurz zu halten. Die Wiederherstellungsmechanismen sollten möglichst automatisiert ablaufen. Die Wiederherstellung von Daten kann z.B. über **Backups** geschehen.

Die Wiederherstellung von Soft- und Hardwarefunktionen wird durch **Hot-Stand-By** Komponenten unterstützt. Sie unterscheiden sich von gespiegelten Komponenten insoferne, als dass sie während des Normalbetriebes nur auf Bereitschaft laufen (und damit für Load-Balancing nicht verfügbar sind) und ausschließlich bei Fehlerfällen aktiv werden. Dazu zeichnen sie während des Normalbetriebes Wiederherstellungsinformation auf und übernehmen Zustandsinformationen der Primärkomponenten. Auf Basis dieser Information versucht die Hot-Stand-By Komponente nach einem Ausfall der Primärkomponente, die Arbeit zu übernehmen, ähnlich der Vorgänge beim Mirroring.

2.6.3. Performance und Skalierbarkeit

2.6.3.1. Performance

Bei der Bestimmung des Begriffes "Performance" und dessen Messung muss im Zusammenhang mit multiuserfähigen Client-Serversystemen, wie sie bei ASP der Fall sind, von anderen Voraussetzungen als beim Betrieb eines Stand-Alone-Systems ausgegangen werden.

ASP-Performancebestimmung und -messung umfasst im wesentlichen drei Komponenten:

- Server
- Client
- Netz

Definiert man Performance als "Geschwindigkeit", so bleiben zwei Sichtweisen, um Server, Netz- und Clientgeschwindigkeit zu messen. Zum einen ist das die maximale Anzahl von Tasks oder übertragbarer Datenpakete pro Zeiteinheit (**troughput***)*, zum anderen die Dauer für die Abarbeitung eines solchen Tasks oder einer Netzwerkanforderung (**response time** oder **execution time**). Besonders im Hinblick auf die SLAs erscheint außerdem eine Unterscheidung in minimale, durchschnittliche und maximale Performance wesentlich, und welche Parameter auf diese Anhaltspunkte Einfluss haben. Zuletzt ist Performance nicht allein ein technischer Begriff, sondern hat auch mit subjektivem Empfinden zu tun.

So kommt es, dass die die Applikation benutzenden Personen, die IT Manager bei den ASPs und die Netzwerkmanager ihren Fokus jeweils auf unterschiedliche Komponenten werfen. Für den ASP ist es wichtig, dass die serverseitige Software effizient mit Ressourcen umgeht und ein Höchstmaß an Performance aus der Hardware herausholt, damit er die Kosten und den Aufwand für diese gering halten kann. Möglicherweise wird der ASP auch, wenn er dies in seinem Geschäftsmodell so vorgesehen hat, dem Ziel entgegenstreben, möglichst viele Kunden gleichzeitig mit Applikationen versorgen zu können (Dies entspricht der Optimierung auf *throughput* und verweist eher auf den Begriff der *Skalierbarkeit*). Ziel sollte aber sein, durch ausreichende Applikationsperformance möglichst viele Endkunden zufriedenzustellen. Per Definition ist es nämlich der ASP Kunde, der eine ASP-Applikation (mehr oder weniger hoch) interaktiv benützt. (Die Performance von anderen Komponenten, die auf das Benutzerempfinden keinen merkbaren Einfluss haben, z.B. die Performance des Backupprozesses, soll außer acht gelassen werden.). Daher soll im ersten Schritt die effektive Performance pro Benutzer analysiert werden. Die Gesamtperformance eines Servers ist eher im Zusammenhang mit seiner Skalierbarkeit in Bezug auf die Anzahl der gleichzeitigen Benutzer und deren Ressourcengebrauch interessant.

Die benutzerbezogene Performance einer ASP-Lösung ist schwieriger zu bestimmen, als die Performance der einzelnen Komponenten. Zum Beispiel findet die Applikationsaktivität nicht am Gerät des Endbenutzers statt, obwohl sie dort empfunden wird.

Im folgenden sind die für ASP und Endkunden wichtigsten Aspekte von Performance genannt.

An erster Stelle steht das eingangs erwähnte subjektive Empfinden von Performance:

- **Benützbarkeit**

  Antwortzeit und wahrgenommene Zeit der ablaufenden Programmschritte bei typischen Arbeitsaufgaben. Erreicht werden soll ein Performancegefühl wie auf einem Desktop-PC. Gemessen wird sie als Zeit zwischen dem Absetzen der Arbeitsaufgabe (z.B. Mausklick) bis zum sichtbaren und merklichen Abschluss derselben. (entspricht der *response time*).

Im Kapitel "Benutzerfreundlichkeit" (2.7.2) werden unter "Performance" die Anforderungen des subjektiven Empfindens an die Performance näher erläutert.

Zu den folgenden technischen Aspekten von Performance lässt sich allgemein sagen, dass Performance in dem Sinne zu verstehen ist, dass alle Ressourcen möglichst effektiv genutzt werden, aber nicht überlastet werden:

- **Effiziente Nutzung des Netzes**

  Aufgaben sind die Minimierung des Datenvolumens und der Anzahl zu übertragender Datenpakete. Dieser Punkt ist wichtig, um die Anforderungen an den NSP bzw. an das Netz zu bestimmen bzw. möglichst gering zu halten. Sinnvolle Messungen sind:

  - **Totalanzahl übertragener Bytes pro Zeiteinheit** (z.B. KB/s) von und zum Server. Die Obergrenze ist erreicht, wenn das Netz ausgelastet ist.

  - **Anzahl Pakete pro Zeiteinheit** von und zum Server. Je interaktiver eine Applikation oder ein Teil derselben ist, desto mehr Pakete müssen übertragen werden. Dieses Faktum kann aufgrund erhöhter Paket-Kollisionen zu einer Verringerung der Bandbreite führen.

- **Clientseitige Ressourcennutzung**

  Herausforderungen sind die Minimierung des clientseitigen CPU-Zeit- und Speicherbedarfs. Dadurch kann ein Client kostengünstige Hardware einsetzen. Gemessen werden kann sie an CPU Auslastung und Speicherbedarf. Diese Werte lassen sich unter bestimmten Betriebsystemen (z.B. Windows NT und UNIX) einfach messen.

- **Serverseitige Ressourcennutzung**

  Ein wichtiger Punkt ist die Minimierung des serverseitigen CPU-Zeit- und Speicherbedarfs und das Management dieser Ressourcen bis zur maximal möglichen Anzahl an Benutzern (Skalierbarkeit).

Um den Einfluss der einzelnen Komponenten auf die Performance näher zu untersuchen, sei im Folgenden erklärt, was beim Absetzen eines Client-Befehls (z.B. Mausklick) im Hintergrund passiert und welche Zeitverzögerungen dabei entstehen:

- **Verzögerung auf Clientseite im Zuge einer Interaktion mit der Applikation**

  Die Client-Anforderung (bei websprachenbasierten Systemen: Mausklick, Eingabeabschluss, bei terminalbasierten Systemen zusätzlich: jeder Tastendruck, jede Mausbewegung) wird registriert, verarbeitet und als Internetanforderung an den Server formuliert. Vorausgesetzt, der Clientrechner erfüllt die Systemvoraussetzungen des Browsers oder des Clientkontrollprogramms, geschieht dieser Vorgang in jedem Fall extrem schnell und vom Benutzer unbemerkt, da er kaum Rechenzeit in Anspruch nimmt. Es wird lediglich die Eingabeinformation (Mauskoordinaten, eingegebenes Zeichen, Objekt-ID des geklickten Buttons, etc.) gemäß einem Internet-Protokoll (z.B. HTTP, SOAP, RDP oder ICA) formuliert. Auch das Sichtbarmachen des Clientbefehls (z.B. gedrückter Button) nimmt i.A. keine merkbare Rechenzeit in Anspruch. Lediglich bei komplexeren GUI-Elementen, die zudem noch selbst einer Serverinteraktion bedürfen (z.B. grafische

Javascript-Pulldownmenus), kommt es zu Verzögerungen zwischen Benutzeranforderung und tatsächlicher Anforderung an die Applikation über das Netz.

- **Verzögerungen bei der Datenübertragung über das Netz**

Gemäß den Anforderungen des darunterliegenden Netzwerkprotokolls (in den meisten Fällen wird dies TCP/IP sein) werden die Daten über das Netz übertragen.

Läuft die Übertragung über das Internet, entstehen dieselben Netz-Verzögerungen wie bei jeder "gewöhnlichen" Internet-Anforderung. Hier zu nennen wären die physikalischen Übertragungsverzögerungen durch die Kabeln, die Verzögerungen bei Repeatern, Switches und Routern, Gateways und Firewalls, und der Overhead durch Domainnameserver-Lookups, wiederholte Übertragungen von Datenpaketen aufgrund von Kollisionen oder Übertragungsfehlern. Gemäss dem TCP/IP Modell sind hierbei die OSI-Schichten eins bis vier (Physikal-, Data link-, Network- und Transportlayer) relevant. Ausführliche Diskussionen über die Performance von TCP/IP sind in [Dunigan 2002] zu finden, sodass hier nicht näher darauf eingegangen werden soll. Jedoch stellen insbesondere terminalbasierte Systeme neue Herausforderungen an die **Netzqualität**: Die Interaktivität ist bei dieser Basistechnologie höher als bei den Websprachen, sodass Mindestqualitäten an Performance an Netze gestellt werden sollten, über die eine ASP Anwendung gefahren wird. Der Bedarf an Hochleistungsnetzen, insbesondere solche mit "**Quality of Service**"-(QoS)-Garantien, wird durch ASP gesteigert. Die Benützbarkeit sollte sich aber auch bei steigender Benutzeranzahl nicht negativ verändern. Die Hauptanbindung (sog. "Backbone") zum ASP (oder NSP) sollte bis zur gewünschten Maximalanzahl an Benutzern skalieren. Die technische Analyse von Topologie und Performance solcher Netze ist jedoch nicht Teil dieser Diplomarbeit, zumal es keine "Netz-Universallösung" für ASP gibt. Im allgemeinen liegt hier auch nicht der Flaschenhals der Netzperformance, sondern er ist im Internet und in den Leitungen zum Endgerät zu suchen ("**last mile**"). Angenommen, ein ASP-Anwender ist über einen beliebigen NSP mit dem Internet verbunden, der nicht notwendigerweise eine direkte Verbindung zum ASP hat. Auf diesem Netz ist die Performance von der Anzahl der Benutzer pro ASP weitgehend unabhängig, da diese nur einen geringfügigen Anteil an der Gesamtlast des weltweiten Internets ausmachen. Da das Performanceverhalten des Internets weitgehend unbeeinflussbar ist und (noch) keine QoS-Performancegarantien für das Internet existieren, kann für einen solchen ASP-Betrieb keine dezidierte Mindestperformance garantiert werden.

Für viele ASPs besteht deshalb die Lösung darin, entweder selbst entsprechende **Direktleitungen** zum Kunden aufzubauen (Standleitungen oder MANs) oder ihre ASP-Software nur über vom ASP bestimmte High-Speed-NSPs zur Verfügung zu stellen, die mehrere Einwahlknoten anbieten. Es gibt ASPs und AIPs, die ASP-Lösungen nur über ihre Standleitungen betreiben und nicht über das Internet, um die in den SLAs genau festgelegten Qualities of Service, wie Mindestantwortzeiten, auch garantieren zu können [Paiha 2001]. Der Nachteil in dieser Lösung besteht darin, dass dadurch die Ortsunabhängigkeit von ASP gefährdet wird. Zum Beispiel ist vom Ausland aus eine Einwahl in die gewünschte Applikation nicht möglich. Um die Performance von ASP zu bewerten ist deshalb auch immer unbedingt eine Miteinbeziehung des Standortes von ASP und Benutzer erforderlich.

Bei webbasierten Diensten ist die Netzgeschwindigkeit wesentlich unkritischer: Üblicherweise werden durch eine Browsersprache eine Reihe von Eingaben zuerst lokal verarbeitet, bevor eine Interaktion mit der serverseitigen Applikation eingeleitet wird. Während bei terminalbasierten Diensten schon allein durch Mausbewegungen ein ständiger Hin- und Rückfluss in Form von TCP/IP Paketen anfällt, gibt es bei webspachenbasierten ASP-Lösungen in unregelmäßigen Zeitabständen Netzpausen mit anschließenden, relativ

umfangreichen Datenübertragungen [Paiha 2001]. Die Netzbelastung ist trotzdem bei webbasiertem ASP wesentlich geringer, weil erstens der Overhead an Steuerungsdaten durch die gestaffelte Datenübertragung geringer ist und zweitens einige Informationen (z.B. Mauskoordinaten) erst gar nicht übertragen werden.

- **Verzögerungen am Server durch Verarbeitung der einlangenden Nachricht, Ausführung der jeweiligen Applikationsfunktion, Vorbereitung der Antwort-Nachricht und Absenden dieser an den Client**

  Im folgenden wird davon ausgegangen, dass eine einem bestimmten Benutzer zugeordnete Applikationssession zu einer Zeit immer nur auf einem Server läuft. Das bedeutet, dass Datenpakete vom Client während einer Applikationssession immer nur an einen bestimmten Server einlangen. Dies ist wichtig herauszustreichen, da sonst die serverseitige Performancemessung wesentlich komplexer wäre.

  Je nach Servertyp werden eingehende Datenpakete zuerst in eine Queue gestellt und anschließend nach ausgewählten Kriterien geschedult (Zeit, Priorität).

  Die Server-Performance hängt stark davon ab, welche Aufgaben außer ASP der Server und die Serversoftware noch haben. Da webbasierte ASP-Dienste fast immer auf HTTP basieren, wird dabei oft derselbe Server für den Webauftritt und die ASP-Dienste benutzt. Durch den bei den webbasierten Diensten meist niedrigeren Interaktionslevel nimmt man längere Verzögerungen durch Server-Last eher in Kauf als bei den terminalbasierten Diensten. Zusätzlich gibt es bei webbasiertem ASP fast immer eine Datenbankanbindung, meist auf einem eigenen Server. Dies macht die Performanceanalyse und -messung äußerst komplex.

  Die Hersteller von Terminalservern empfehlen nicht, auf demselben Servergerät, auf dem der Terminalserver läuft, zusätzlich andere Internet-Dienste zu installieren, da es dabei z.Z. noch das Problem gibt, dass manche Applikationsprozeduren den Server zu 100% auslasten können [Paiha 2001].

2.6.3.2. Skalierbarkeit

Skalierbarkeit ist die Fähigkeit von IT-Systemen, kontinuierliche Performance bei steigendem Ressourcenverbrauch zu gewährleisten.

Um ein skalierbares System zu erhalten, müssen unterschiedliche Maßnahmen gesetzt werden, um einerseits den Server bei steigender Benutzeranzahl nicht zu überlasten, d.h. die Gesamtperformance auf dem gewünschten Niveau zu halten und andererseits die Antwort- und Ausführungszeiten der Applikationsfunktionen pro Anwender niedrig zu halten.

Die Auslastung der ASP-Infrastruktur (Hard- und Software für Server und Netz) wird weitgehend von zwei Parametern beeinflusst:

- **Anzahl der gleichzeitig arbeitenden ("eingeloggten") Benutzer**

  Jeder zusätzliche Anwender erfordert Verwaltungsaufwand auf Serverseite. (Sessionverwaltung, Ressorcenverwaltung). Bei einer geclusterten Serverarchitektur muss die gesamte Server-Farm in Bezug auf die Anzahl der Server bzw. jeder Server in Bezug auf seine Ressourcen skalierbar sein. Grenzen kann hier bereits das Betriebsystem setzen: Windows 2000 Datacenter Server erlaubt den Einsatz von maximal 16 CPUs und maximal 64GB Hauptspeicher pro Servergerät. Ebenfalls muss die Netzwerkstruktur bei steigender Benutzeranzahl skalieren. Im Hinblick auf die SLAs muss auf steigende Netzlast im Backbone-Bereich mittels einer Steigerung der Netzkapazität, z.B. durch Hinzunahme eines weiteren NSPs oder ISPs reagiert werden.

Beim Log-In wird jeder Anwender einem bestimmten Server zugeteilt (bei websprachenbasierten Systemen erfolgt die Zuteilung u.U. bei jeder Transaktion). Während bei websprachenbasierten Systemen u.U. mehrere hundert bis maximal tausend Benutzer pro Server gleichzeitig arbeiten können, sind es bei dem Terminalsystem Citrix MetaFrame XPe nur 40 bis maximal 100. Auch setzt das zentrale Management in Bezug auf die maximale Anzahl an Servern pro Server-Farm Grenzen. Diese sind jedoch mit etwa 1000 Servern bei Applikationsserverprodukten (auf Websprachenbasis) und Terminalserverprodukten sehr großzügig bemessen.

- **Ressourcenverbrauch pro Anwendung und Anwender**

Betrachtet man die Server- und Netzbelastung für eine individuelle Benutzersession, so fällt auf, daß insbesondere auf Skalierbarkeit im Back-End-Bereich (Datenbanken und Dateisysteme) Rücksicht genommen werden muss. Es gibt die "Power-User", die besonders oft und intensiv Operationen auf Daten (viele Schreiboperationen) durchführen, und andererseits Benutzer, die nur ab und zu mit der Applikation interagieren.

Auf der anderen Seite gibt es bestimmte Anwendungsfunktionen, die besonders viel Ressourcen benötigen. Die Rechtschreibprüfung von MS Office benötigt z.B. relativ viel Prozessorressourcen, das Drucken über Terminalsysteme ist sehr netzressourcenintensiv.

Der Netzressourcengebrauch auf der "Last-Mile" ist hingegen bei beiden Basistechnologien in Bezug auf die Präsentation der Benutzerschnittstelle relativ konstant und übersteigt auch bei intensivem Gebrauch nicht einen bestimmten Wert. Deshalb ist bei einer entsprechend schnellen Front-End-Netzanbindung (ADSL, Kabelmodem, Standleitung) dessen Skalierbarkeit eher von geringer Bedeutung. Von den langsameren Einwahlverfahren (z.B. Modem, ISDN) ist jedoch nur ISDN mittels Kanalbündelung skalierbar.

An das Endgerät werden bei ASP nahezu keine besonderen Anforderungen an Skalierbarkeit gestellt.

2.6.4. ASP Operation Management

Die bisher vorgestellten Aufgaben eines ASPs und dessen Technologien verlangen nach einer effektiven Organisation.

Ziel des Operation Managements bei ASP ist, die Qualitätssicherung, die Kosteneffizienz, die reibungslose Arbeit und Zusammenarbeit aller ASP-Player und eine Differenzierung zu den Markt-Mitbewerbern herzustellen und auf bestimmte Zeit sicherzustellen.

Die Aufgaben in diesem Bereich sind sehr vielfältig, komplex und umfangreich. Dabei werden IT-Geschäftsprozesse des ASPs geplant, durchgeführt und dokumentiert.

Das Operation Management durchläuft nicht einen konkreten Schritt-für-Schritt Plan, sondern bedeutet eine ständige Anpassung und Überprüfung der Vorgänge.

Für den ASP und seine Partner bedeutet dies die Zuteilung der Aufgabenbereiche an die Mitarbeiter, die Erstellung eines Service-, Security- und Kostenplans, die Definition von Verträgen und Service Level Agreements, die Erlangung von Softwarelizenzen, die Einrichtung von Qualitätsmanagement, Risikoabschätzung und des Verrechnungsmanagements, die Analyse und Sicherstellung der Kundenzufriedenheit und die lückenlose Dokumentation all dieser Aufgaben [Infonomics 2001], [CompTIA 2001].

Ein anderer Teil der Operation Management-Prozesse findet ihre Abbildung in computerunterstützte Vorgänge, die mit dem ASP-Serverbetrieb integriert sein müssen. Als Schnittstelle zu den Infrastrukturkomponenten dienen standardisierte Protokolle (HTTP, XML, Log-Dateien etc.). Manche dieser technischen Vorgänge erfordern ihrerseits wiederum eine geeignete Organisation und Verwaltung.

Die halbautomatische softwaretechnische Unterstützung beim ASP Operation Management konzentriert sich auf folgende Bereiche:

- Application Management
- Delivery Management
- User Management

Die Software für diese Problembereiche ist nicht in die ASP Basistechnologie integriert. Es gibt Spezialapplikationen. Hersteller sind z.B. IBM (Tivoli), Xevo, OSI, BMC und Marimba.

Diese drei Bereiche von Service Management Software sollen nun genauer betrachtet werden.

2.6.4.1. Application Management

Dieser Aufgabenbereich lässt sich auch als "Wartung" beschreiben. Es handelt sich dabei um Installation, Upgrade, Konfiguration und Freischaltung von Soft- und Hardware. Da Wartungen nicht ständig während des ASP Betriebs anfallen, sondern nur zu den genau geplanten Wartungszeiten, werden diese Aufgaben halbautomatisch über interaktive Benutzereingriffe getätigt.

- **Lizenzmanagement**

  Das Lizenzmanagement hält Lizenzinformationen für alle internen Produkte und angebotenen Dienste des ASPs bereit. Hier werden die Zugehörigkeit, die Gültigkeitsdauer und die Konditionen der Lizenzen verwaltet.

- **Release Management**

  Dieser Prozess beinhaltet die Anforderungen an Hard- und Software sowie Dokumentation für die Freischaltung eines ASP-Dienstes. Eine entsprechende Versionskontrolle erlaubt eine konsistente Umgebung vor der Freischaltung. Die ASP Managementsoftware unterstützt durch entsprechende Testumgebungen und Versionskontrolle.

- **Change Management**

  Change Management ist ein zentraler Prozess, der alle Change Requests zusammenführt und sicherstellt, dass kontrollierte Entscheidungen über Change Prozesse getroffen werden. Das Change Management Softwaremodul empfängt Change Requests vom Service Level Management (manuell oder automatisiert), von der User Management Einheit (Kundenwünsche) und vom Delivery Management (Problem Management).

  Wenn z.B. Updates oder neue Installationen durchgeführt werden, informiert das Service Level Management (laut Planung in den SLAs) das Change Management Modul.

  Wenn neue SLAs aus Gründen veränderter Geschäftsbedingungen oder Einführung von neuen Services angewendet werden, müssen solche Veränderungen dokumentiert und archiviert werden. Auch Veränderungen aufgrund von Kundenwünschen bzw. -beschwerden fließen hier ein. Schließlich werden unvorhersehbare Veränderungen im System (z.B. Ausfälle) über das Problem Management Modul erkannt.

- **Configuration Management**

  Hier werden technische Veränderungen an Soft- und Hardware und dessen Spezifikationen und Konfigurationen festgehalten. Dieses Modul befasst sich mit der Vergabe der Berechtigungen sowie der Kontrolle, Durchführung und Dokumentation von Konfigurationsänderungen. Mittels Konfigurationsmanagement werden die Ergebnisse der Change Requests umgesetzt und teilweise an die Kunden mittels User Management weitergeleitet.

2.6.4.2. Delivery Management

Delivery Management befasst sich mit allen Aufgaben der Bereitstellung der Applikationsdienste und Zusatzleistungen. Dieser Aufgabenbereich ist besonders kritisch, da er das Management des eigentlichen ASP Betriebs darstellt und aus diesem Grund ständig (zu den vereinbarten Uptime-Zeiten) aktiv sein muss. Aus diesem Grund arbeiten die Softwarekomponenten beim Delivery Management weitgehend automatisiert.

Die Hauptaufgaben sind:

- **Service Monitoring**

  Das Aufzeichnen sämtlicher technischer Aktivitäten der Infrastruktur (Server, Server Plattform (Betriebsystem), Applikationen, Netz) ist einerseits eine Verpflichtung des ASPs, um die SLAs nachweisen zu können und andererseits ein geeignetes Mittel, Engpässe und Fehler zu registrieren und darauf aufbauend Ausfälle rechtzeitig zu vermeiden und Verbesserungen anzustreben. Die Service Monitoring Einheit ist mit der Report- und der Problem Management Einheit verbunden.

  Ressourcen innerhalb der Infrastruktur sind Applikationsserver, Datenbanken, Dateiserver, Storage-Lösungen, Netzwerkkomponenten (Router, Switches, Kabel; kurz, die gesamte Netzverbindung vom Kunden bis zum ASP) und die Gesamtheit aller serverseitigen Software, inklusive der ASP-Applikation selbst.

  Dies ist ein Auszug der Daten, die während des Betriebs aufgezeichnet werden (müssen):

  - Log-In- und Log-Out Zeiten
  - Benutzerkennungen
  - Dauer der Verbindungen
  - Status der Verbindungen
  - Auslastung der Ressourcen
  - Performance der Ressourcen (Throughput, Response time)
  - Details über Ausfälle und Fehlfunktionen der Ressourcen
  - Verfügbarkeit (MTTF, MTTR) der Ressourcen
  - Dateizugriffsprotokolle
  - Datenbankprotokolle

  Auf Basis dieser Daten errechnet die Service Management Software für die einzelnen Kunden Vergleichswerte für die in den SLAs definierten Service Parameter.

- **Report und Response Management**

  Diese Einheit ist mit der Service Monitoring Einheit und mit der User Management Komponente verbunden.

Auf Basis der Monitoring-Daten werden Ursachen und deren verantwortliche Komponenten bei Fehlern und Warnsignalen im System aufgespürt.

Es werden High-Level-Reports kundenspezifisch generiert und an die User Management Komponente weitergeleitet.

In die Reports werden auch nicht-technische Ereignisse und Problemfälle aufgenommen, soweit sie für den Kunden oder ASP relevant sind, wie z.B. Service Level Agreement Verletzungen und Kosten.

- **Problem Management**

Ist bereits ein Fehler (d.h. ein unbeabsichtigtes Ereignis) im System aufgetreten, muss der ASP möglichst schnell und automatisiert reagieren, um die SLAs einhalten zu können.

Bei Service Delivery Problemen müssen die Schritte identifiziert werden, die zur Lösung und zukünftigen Vermeidung notwendig sind. Insbesondere ist zu achten, dass sich "Fehler im System" möglichst wenig auf den Kunden auswirken.

Je nach Schweregrad des Fehlers sollte das computerunterstützte Problem Management selbständig entscheiden, ob zur Lösung des Problems ein Benutzereingriff notwendig ist und dem ASP entsprechend behilflich sein.

Kann die Problem Management Einheit eine Verletzung von SLAs nicht mehr abwenden, veranlasst sie über die User Management Komponente den Kunden darüber zu informieren. Sie veranlasst außerdem eine entsprechende Änderung der Service Regeln ("Strafen" für den ASP wie z.B. reduzierter Preis des Applikationsdienstes für eine bestimmte Zeit).

Die Wiederherstellung des ordnungsgemäßen Betriebes nach einem Ausfall wird auch "Disaster Recovery" genannt.

Für schwerwiegende Szenarios ist meist ein sog. "Escalation Management" implementiert, das jedoch nicht ohne Zuhilfenahme menschlicher Akteure auskommt.

- **Security Management**

Das Security Management regelt auf Basis der Security-Policy und der SLAs den sicheren Zugang zum Applikationsdienst. Die Voraussetzungen für eine sichere Architektur müssen jedoch erst von Sicherheitsexperten durch Implementieren eines geeigneten Sicherheitsplans (Security-Policy) geschaffen werden. Der Security Management Einheit kommt also eine durchführende und kontrollierende Funktion zugute. Sie ist weiters verantwortlich für die Erkennung von potentiellen oder eingetretenen Sicherheitsverletzungen und deren Einstufung in den Schweregrad.

Die Aufgaben eines ASPs im Sicherheitsbereich werden in einem eigenen Kapitel (Kapitel 2.6.5.) behandelt.

- **Verrechnungsmanagement**

Auch diese Verwaltungseinheit muss während der gesamten Betriebsdauer aktiv sein. Sie ist einerseits mit der Service Monitoring Einheit und andererseits mit dem User Management verbunden, manchmal auch mit den Applikationen selbst ("Pay per use"). Die Architektur und Anforderungen eines Billing Managements wurden aufgrund der starken Relevanz im ASP-Bereich als eigene Herausforderung erkannt und in einem eigenen Kapitel behandelt. Denn nur ein effektives Billing-System schafft transparente und verlockende Angebote für ASP Kunden.

2.6.4.3. User Management

Das User Management befasst sich mit allen benutzer- und kundenbezogenen Aspekten ("Customer Care").

- **Help Desk**

  Der Help Desk ist die zentrale Anlaufstelle für den Kunden bei allen Anliegen und Problemen. Sie kann in Form eines Call-Centers (auf Basis von Telefoniediensten) realisiert sein und/oder auf Internetbasis. Obwohl meist als eigene Applikation realisiert, arbeitet sie doch stark mit der Operation Management Software zusammen.

  Aufgaben sind die Archivierung von Problemfällen samt deren Lösungen, die Klassifizierung von Problemen und Beschwerden und die webbasierte Integration von Operation Management Daten zur Problemlösung. Dem Help Desk kommt eine zentrale Rolle im Operation Management zu, da sich die Kunden eine rasche Beantwortung von Fragen und Lösung von Problemen erwarten.

- **Account Management**

  Hier werden alle Benutzerdaten und Benutzerberechtigungen in einer Datenbank festgehalten. Falls gewünscht, bietet die Account Management Software auch ein webbasiertes Front-End für den Kunden zur Administration seines Zuganges an. Dies stellt quasi einen zusätzlichen Applikationsdienst dar.

- **Service Reporting**

  Die Service Reporting Einheit empfängt Daten vom Report Management und dem Verrechnungsmanagement. Hier werden diese Daten aufbereitet und in eine für den Kunden lesbare Form formatiert. Die Service Reporting Einheit versendet dann die Berichte auf dem gewünschtem Weg (E-Mail, Web, Briefverkehr).

- **Self Support**

  Self Support ist eine sinnvolle Einrichtung zur Entlastung des ASP und trägt bei einfachen Problemen zu einer schnelleren Problemlösung bei. Self Support ist ebenfalls webbasiert und kann, falls vorhanden, das Account Management integriert haben. Da man bei ASP ohnehin online ist, ist dieser Weg manchmal schneller als der Griff zum Telefonhörer. Man unterscheidet:

  - **Self Help** - bietet computerunterstützte, automatisierte Hilfestellungen
  - **Self Diagnosis** – hilft einem Benutzer die Ursache für ein Problem zu finden und
  - **Self Healing** – ermöglicht Kunden, selbst Fehler zu beseitigen

Aus der kurzen Vorstellung von ASP Operation Management ist ersichtlich, dass sich hier eine der größten Herausforderungen im gesamten ASP Bereich stellen. Mit ITIL (IT Infrastructure Library) wurde aus diesem Grund ein Standard geschaffen, der Service Provider beim Operation Management unterstützt. Zudem findet sich in [CompTIA 2001] eine detaillierte Spezifikation für "best-practices" im elektronischen Dienstleitungsbereich, die nicht nur für ASPs geeignet ist, sondern für alle IT-Service Provider eine Anweisung für effektives IT Management darstellt.

2.6.5. Sicherheit

Sicherheit ist eine der bedeutensten Angelegenheiten sowohl für den ASP als auch für dessen Kunden. In einer Studie von Zona Research im Oktober 2000, die für das ASP Industrie Konsortium erstellt wurde, stuften 96% der tatsächlichen und potentiellen ASP-Kunden Sicherheit als entweder "extrem wichtig" oder "sehr wichtig" ein [Neefe 2000].

Der Kunde übergibt im Zuge des ASP-Geschäftsverhältnises dem ASP die vertrauensvolle und verantwortungsvolle Aufgabe, für die Geheimhaltung, Integrität und Verfügbarkeit von unternehmenseigenen, geschäftskritischen Applikationen und Daten zu sorgen.

Dabei unterliegt das ASP-Modell grundsätzlich den Sicherheitsrisiken jedes IT-Umfeldes. IT-Sicherheit ist nicht mit einem einzigen Produkt herzustellen, sondern erfordert technische, organisatorische und soziale Maßnahmen, die in einem **Sicherheitsplan** festgehalten werden ("Security Policy"). Teilweise sind Sicherheitspläne und -maßnahmen sogar gesetzlich vorgeschrieben. Dies alles geschieht in Abstimmung mit den vertraglich zugesicherten Service Level Agreements (SLAs). Erfolgreiche ASPs wie Corio, USInternetworking oder Interliant präsentieren ihr Sicherheitskonzept transparent ihren Kunden.

Die Abhängigkeit von unternehmenskritischen Applikationen und Daten sowie die Geheimhaltung derselben, die grenzenlose, offene und grundsätzlich ungesicherte Beschaffenheit des Internets sowie die gemeinsame Nutzung von Internet, ASP-Infrastruktur und Applikationen durch mehrere Nutzer stellen neue, zusätzliche Herausforderungen dar.

Eine gute Möglichkeit für ASPs, diese Aufgaben zu bewerkstelligen, stellt die Zusammenarbeit und Beratung mit IT-Sicherheitsexperten dar oder das Auslagern der gesamten ASP-Sicherheitslösung an einen sog. Security Provider.

Allerdings können ASPs in der Praxis keine hundertprozentige Sicherheit garantieren. Man muss jedoch zugestehen, dass eine totale Sicherheit ohne die Kooperation mit einem ASP genauso wenig existiert [Wagner 2001].

2.6.5.1. Elemente der Sicherheit

Der Begriff "Sicherheit" umfasst in der Informationstechnologie bestimmte Elemente, die dazu dienen, den Schutz von Daten, Soft- und Hardware, Interessen und Know-how von ASP und ASP-Kunden zu wahren. Eine umfassende Sicherheitslösung muss all diese Aufgaben bewältigen.
Die folgenden Begriffe beschreiben, was unter IT-Sicherheit zu verstehen ist.

- **Authentifizierung**

  Die Authentifizierung ist dazu da, die Identität eines Benutzers, sei es eine Person, ein automatisierter Prozess oder eine andere Anwendung, durch geeignete Authentifizierungsmaßnahmen (Benutzername-Passwort-Kombination, digitale Unterschrift oder physikalische Attribute wie z.B. Fingerabdruck) festzustellen und den Zugriff für nicht berechtigte Benutzer zu verweigern. Die Anforderungen an Angaben zur Identität können von Synonymen bis zu digitalen Unterschriften und genauen Personenangaben reichen (wichtig z.B. für die Verrechnung).

- **Zugriffskontrolle**

  Die Zugriffskontrolle hat en Zweck, die Berechtigung für die Benutzung von Diensten und Daten zu verifizieren und eine unbefugte Nutzung zu vermeiden. Die Zugriffkontrolle regelt mittels sog. Rechte die Art und das Ausmaß von Zugriffen auf Server-Ressourcen wie

Applikationsfunktionen, Daten und Verzeichnisse für jeden einzelnen authentifizierten Benutzer oder authentifizierte Benutzergruppe.

- **Geheimhaltung**
  Wenn vertrauliche Daten (z.B. Kreditkartennummern, Geschäftsgeheimnisse oder persönliche Daten) zwischen Kunde und ASP transportiert werden, so besteht der Wunsch nach Geheimhaltung. Die Daten müssen solcherart geschützt werden, dass sie nicht von unbefugten Dritten eingesehen oder manipuliert werden können. Im Rahmen der Netzsicherheit kann die Datenübertragung z.B. physikalisch auf die Weise geschützt werden, dass schwer auszuspähende Medien (z.B. Lichtleiter) benutzt werden. Durch logische Sicherheitsmaßnahmen auf Ebenen der Authentifizierung und Zugriffskontrolle kann ein gewisser Schutz von vertraulichen Daten hergestellt werden. Einen umfassenden Schutz von vertraulichen Daten vor "Lauschangriffen" bietet jedoch erst die **Verschlüsselung** (siehe "Verschlüsselungskonzepte und -maßnahmen).

- **Integrität**

  Integrität bezeichnet die Unverfälschtheit von Daten. Daten dürfen nur von befugten Personen modifiziert werden. Integrität umfasst auch die Unverfälschtheit von zwischen ASP und Kunde zu übertragenen Daten über das Internet. Integrität von Internet-Kommunikation kann mittels **Prüfsummenverfahren** (z.B. MD5), meist in Kombination mit Verschlüsselungsverfahren und digitalen Unterschriften, garantiert werden.

- **Verfügbarkeit**

  Die Verfügbarkeit befindet sich in einem Grenzbereich der IT-Sicherheit. Nur wenige Aspekte der Verfügbarkeit werden tatsächlich zur Sicherheit hinzugerechnet. Im allgemeinen sind davon nur durch Angriffe (typischer Angriff: "Denial of Service", siehe "Sicherheitsbedrohungen") ausgelöste Beeinträchtigungen der Verfügbarkeit (z.B. Netzausfall, Serverausfall, Applikationsausfall) betroffen. Zur detaillierten Darstellung von "Verfügbarkeit" siehe Kapitel 2.6.2.

- **Nicht-Leugnung**

  "Nicht-Leugnung" bezeichnet die garantierte, nachweisbare und identifizierbare Übertragung von unverfälschten Daten. Dies beinhaltet, dass der Sender einer Nachricht dessen korrekte oder inkorrekte Zustellung nachweisen kann. Außerdem teilt der Sender dem Empfänger seine Identität mit, sodass keiner die Verarbeitung der Daten leugnen kann. "Nicht-Leugnung" wird durch Zertifikate und digitale Unterschriften bewerkstelligt.

2.6.5.2. Sicherheitsbedrohungen

Sicherheitsbedrohnungen können von vielen Richtungen ausgehen. Grundsätzlich müssen die ASP-Infrastruktur und die ASP-Applikationen derart gestaltet sein, dass das System bei bestimmungsgemäßem Gebrauch und Umgang durch ASP Benutzer und ASP Mitarbeiter keine Sicherheitslücken offenbart. Solche Sicherheitslücken stellen im Hinblick auf mögliche Schäden, Schuldzuweisungen und Schadensersatzansprüche ein ernsthaftes Problem dar, da es sich hierbei um unbekannte Sicherheitsbedrohungen handelt, die vom System selbst ausgehen.

Weitaus häufigere als durch versehentliche Fehlbedienungen verursachte Sicherheitsbedrohungen stellen jedoch bewusste, böswillige Angriffe dar. Untersuchungen von Unternehmen [Neefe 2000] zeigen, dass etwa 65% der Sicherheitsverletzungen von unternehmenseigenem Personal ausgehen. Auch Nicht-Administratoren kennen das System und

dessen Sicherheitslücken. Diese Zahl ist allerdings mit Vorsicht zu genießen, da eine bloße Übertragung auf die Untermenge der ASP Unternehmen unzulässig ist. Für die ASP Sicherheit existieren derzeit noch keine Studien bezüglich vorgefallener Sicherheitsangriffe.

Wie auch immer die Verteilung von internen und externen Angriffen aussehen mag, sog. **Hacker** stellen jedoch ein immer ernsthafter werdendes Problem dar.

In einer im Jahr 2000 durchgeführten Umfrage gaben 91% von 643 befragten Unternehmen an, mindestens eine unbefugte Nutzung ihres IT-Systems in den letzten 12 Monaten registriert zu haben, was zu einem akkumulierten Verlust von 626 Mill. US-$ in den Jahren 1997 bis 2000 geführt hat [Hollander 2002]. Für ASPs wird hier ein wesentlich geringerer Betrag erwartet, da die Sicherheit einen elementaren Geschäftsbereich von ASPs darstellt. Es liegen jedoch noch keine frei verfügbaren Studien zur ASP Sicherheit vor.

Informationen für effektives Hacking liegen in immer höherem Ausmaß im Internet vor. Die Motivationen für Hacker können vielseitig sein (z.B. Neugierde, Langeweile, Rache, Betrug, Diebstahl), es erscheint jedoch einsichtig, dass ASPs, insbesondere wenn sie zahlreiche Groß-Unternehmen betreuen, ein potentielles Angriffsziel darstellen. Bezüglich der illegalen Benutzung von ASP Applikationen bleibt jedoch noch abzuwarten, ob hier eine neue Art der Softwarepiraterie entstehen könnte.

Ein vernetztes Computersystem, wie ASP, kann zahlreichen Attacken ausgesetzt sein, die zu unterschiedlich hohen Schäden führen:

- **Eindringung und Spionage**

  Dies sind u.A. Angriffe auf die **Geheimhaltung** von Daten. Unter Umgehung der **Authentifizierung** und **Zugriffskontrolle** (z.B. durch Ausspähen, "Erraten" oder systematisches Ausprobieren von Passwörtern) dringen Hacker in Computersysteme ein. Sie forschen die Server Architektur aus, suchen mögliche Sicherheitslücken und nehmen möglicherweise auch Einsicht in vertrauliche Daten. Der Hacker fertigt sich weiters lokale Kopien (Download) von wertvoller Software und Information an.

  Andere Möglichkeiten zur Spionage bieten z.B. auch ein "Lauschangriff" auf die Datenkommunikation zwischen ASP und Kunden sowie eine unbefugte, mündliche, schriftliche oder sonstwie übermittelte (z.B. durch Einbruch ins Data Center) Weitergabe von Information.

  Da bei dieser Art von Angriffen die Hard- und Software-Integrität nicht beeinträchtigt wird, werden solche Angriffe vom ASP oft gar nicht bemerkt.

  Ein wirksames Mittel gegen Eindringung und Spionage stellen **Firewalls** dar (siehe "Netzwerksicherheit").

- **Diebstahl, Manipulation, Unterbrechung und Einschleusung**

  Diese Angriffe stellen Beeinträchtigungen der **Integrität** des ASP Betriebes dar. Durch Einbruch oder illegalen Zugang zum Data Center können wertvolle Hard- und Softwarekomponenten und Daten gestohlen, manipuliert oder zerstört werden.

  Üblicher sind jedoch Manipulationen auf dem virtuellen Weg über das Internet. Hierbei könnten z.B. Geschäftsdaten von ASP Kunden manipuliert werden, was im schlimmsten Fall katastrophale Auswirkungen für das betroffene Unternehmen haben kann.

  Ein klassischer Fall von Unterbrechung ist die sogenannte "**Denial of Service**" (DoS)-Attacke. Hierbei werden bestimmte Ressourcen, üblicherweise Server, Router oder die Firewall bewusst überlastet oder zum Absturz gebracht. Dies ist zusätzlich ein Angriff auf die **Verfügbarkeit**. Eben genannte Angriffe können mittels einer modernen, effektiven Firewall abgewehrt werden.

  Die bekanntesten Fälle von Einschleusung stellen **Viren** (oder Abarten wie z.B. trojanische Pferde) dar. Viren bestehen aus einem kurzen Stück Programmcode, schleusen

sich in Daten oder Programme ein, pflanzen sich über Netze und Computersysteme fort und können beträchtlichen Schaden anrichten. Viren können mithilfe eines geeigneten **Virenschutzes** verhindert werden.

Hacker schleusen auch ihre eigenen Hilfsprogramme zur weiteren Ausforschung von Daten und der Server Architektur ein, und führen diese Programme auf dem Server aus.

- **Maskierung**

Eine beliebte, weil relativ einfach Möglichkeit für alle möglichen Arten von Angriffen stellt die Maskierung dar. Es bedeutet die illegale Eindringung in ein Computersystem mithilfe der Authentifizierungsdaten einer anderen berechtigten Person. Sehr oft ist diese Art der Eindringung bei der einfachen Authentifizierung über Benutzername und Passwort anzutreffen.

Maskierung kann durch Zertifikate, digitale Unterschriften, biometrische Authentifizierungsverfahren wie Fingerabdruck oder Gesichtsmerkmal-Authentifizierung weitgehend verhindert werden. Allerdings setzen sich die aufwendigen und teuren biometrischen Verfahren bei ASP Kunden nicht durch, sodass durchwegs auf digitale Unterschriften und Zertifikate vertraut wird.

- **Unauthorisierte Transaktionen**

Von "unautorisierten Transaktionen" spricht man, wenn ein Hacker Applikationsfunktionen benützt, ohne dafür befugt zu sein. Die einfachste und häufigste Möglichkeit dies zu tun, ist der Weg über die Maskierung. Oft ist bei dieser Art von Angriffen auch die Integrität bedroht, bei E-Commerce-Anwednungen sogar in hohem Ausmaß. Die durch den Schaden entstandenen Kosten können entweder zu Lasten des ASPs fallen, oder, falls die unautorisierten Transaktionen unbemerkt bleiben und Maskierung im Spiel ist, auch zu Lasten anderer Kunden.

Im ASP-Bereich liegt mit der nicht lizenzierten Nutzung von Software auch der Fall der Softwarepiraterie vor. Eine Abwehr von Angriffen dieser Art kann am effektivsten mit Maßnahmen der Server- und Applikationssicherheit begegnet werden.

- **Missbrauch**

Missbrauch liegt vor, wenn die Nutzung einer ASP-Applikation durch einen grundsätzlich authentifizierten und autorisierten Benutzer über die vertraglich geregelten Befugnisse hinausgeht. Der Benutzer versucht hierbei die Nutzungsmöglichkeiten der ASP Software oder der Applikationsinfrastruktur zu erweitern oder diese in einer unbefugten Art und Weise zu benutzen. Solche Angriffe werden oftmals durch Software-Fehler ("**Bugs**") begünstigt. Ein Beispiel ist das "Aufbohren" von Test-Versionen oder eingeschränkten Versionen von Applikationen zu Vollversionen, ohne für die Vollversion zu bezahlen. Auch dieses Bedrohungsrisiko wird am besten durch Server- und Applikationssicherungsmaßnahmen minimiert.

2.6.5.3. Sicherheitskonzepte und -maßnahmen

- **Sicherheitsplan (Security Policy)**

Zu Beginn jeder Sicherheitslösung steht ein Anforderungsprofil auf Unternehmensebene des ASPs in Form eines Sicherheitsplans [ECO 2001]. Er beinhaltet Konzepte, Regeln, Bestimmungen, Vorschriften, und Maßnahmen in technischen, organisatorischen und sozialen Bereichen, um einen kontinuierlich sicheren IT-Betrieb zu gewährleisten.

Erst nach einer genauen Identifikation, welche Informationen geschützt werden sollen und wie dies geschehen soll, kann ein Sicherheitsplan aufgestellt werden.

Ein Sicherheitsplan reicht von der Definition der unternehmensweiten (ASP-weiten) Regelung von Sicherheitsaspekten bis zu geräte- und applikationsbezogenen Aspekten.

Einige der grundlegenden Fragen, die durch den Sicherheitsplan beantwortet werden müssen, umfassen:

- Welche Information ist vertraulich und muss geschützt werden ?
- Wie wird diese Information geschützt ?
- Wird die Information verschlüsselt ?
- Wird der Zugriff auf Information eingeschränkt ?
- Wer darf auf Information zugreifen oder sie verändern ?
- Wer darf Zugriff auf Information beantragen ?
- Wer darf Zugriff auf Information gewähren ?
- Wer ist berechtigt, die kritische IT-Infrastruktur zu installieren, zu warten und zu konfigurieren (Benutzerprofile, Rechtevergabe, etc.) ?
- Was passiert, wenn ein versuchter oder gelungener Angriff auf die Sicherheit eingetreten ist ?

Ein Sicherheitsplan muss so gestaltet sein, dass er kontinuierlich anpassbar und umsetzbar ist. Fortschritte sowohl bei den Angriffsmethoden als auch bei den Sicherheitsmaßnahmen müssen rasch in der Security Policy abgebildet werden und wirksam umgesetzt werden.

- **Change Management**

Bevor irgendwelche Veränderungen am ASP-System implementiert werden, müssen formale Mechanismen zur Identifizierung der Auswirkungen auf die Sicherheit hergestellt werden und von einem qualifizierten Sicherheitspersonal evaluiert werden [Chiu 2001]. Alle Vorgänge müssen im Sicherheitsplan dokumentiert und geregelt sein. Dazu gehört z.B. unter welchen Bedingungen ein Account angelegt oder gelöscht wird und wie Hardware, Software und Netztopologie gewartet werden. Ein sehr wichtiger Teil des Change Managements betrifft Updates und Konfigurationen. Hier müssen strenge Qualitätskontrollen vorgenommen werden, um zu überprüfen, ob die neue Version oder die neue Konfiguration mit den Sicherheitsvorschriften konform geht. Zur technischen Vorgangsweise siehe "ASP Operation Management".

- **Security Auditing**

Security Auditing stellt einen sehr wichtigen Mechanismus zur Erkennung von Sicherheitsbedrohungen oder -angriffen dar. Ein Teil davon ist das "Intrusion Detection System" (IDS), das meist in die Firewall integriert ist (siehe "Firewall"). Weiters sollte bei allen Komponenten, die die Protokollierung von Ereignissen beherrschen, diese Funktion auch aktiviert sein. Für die automatische Meldung von Sicherheitsangriffen sollten entsprechende Alarmsysteme installiert sein. Neue Verwundbarkeiten von Hard- und Software werden nahezu täglich entdeckt [Chiu 2001]. Entsprechende Vergleichsmuster zur Erkennung von neuen Sicherheitsbedrohungen oder Patches zur Behebung von Software-Fehlern sollten so rasch als möglich installiert werden, um einen maximalen Schutz zu gewährleisten.

- **Single Sign-On**

  Im Zuge einer Applikationssitzung werden u.U. mehrere Module aufgerufen, die einer Authentifizierung bedürfen. Single-Sign-On ermöglicht die Vergabe einer einzigen Berechtigung und eine transparente Weiterreichung zur Netzwerk-, Betriebsystem- oder Applikationsauthentifizierung. Der Benutzer muss sich nur einmal pro Applikationssitzung authentifizieren (oder nach zwischenzeitlichem Sperren der Sitzung) und braucht nur einen Zugangscode (z.B: Benutzername/Passwort) merken oder zu verwalten.

- **Passwörter**

  Die gebräuchlichste Authentifizierungsmaßnahme ist die sog. "two-factor" Authentizierung. Durch Eingabe eines Benutzernamens, kombiniert mit einem Passwort, wird ein Benutzer eindeutig identifiziert. Passwörter bieten nur einen mittelmäßigen Schutz. Das Problem ist, dass Passwörter nur kurz sind, um im Gedächtnis behalten zu werden und somit die Möglichkeit des Ausspähens, Erratens oder Probierens gegeben ist.

  Um die volle Schutzwirkung zu erfüllen, müssen Passwörter im Gedächtnis behalten werden und dürfen nicht niedergeschrieben werden. Passwörter sollten zumindest 8 Zeichen enthalten und keine leicht zu erratenden Begriffe darstellen. In verschiedenen Bereichen (z.B. Banken, E-Mail-Zugang, ASP) sollten unterschiedliche Passwörter benutzt werden. Nach Ablauf eines fix vorgegebenen Zeitrahmens sollte eine Änderung des Passwortes zwingend vorgeschrieben sein.

- **Verschlüsselung**

  Verschlüsselung dient dazu, Information für nicht berechtigte Personen unlesbar zu machen und somit die Geheimhaltung von Information sicherzustellen. Nur diejenigen, die den zur jeweiligen Benutzeridentität passenden Schlüssel (z.B. Passwort) besitzen, können die Information entschlüsseln. Weiters dient Verschlüsselung gemeinsam mit Prüfsummenverfahren auf Daten- und Applikationsebene dazu, die Integrität sicherzustellen und dezidierte Manipulationen (z.B. Aufhebung der Authentifizierung) zu erschweren. Wenn dem Hacker der entsprechende Schlüssel fehlt, kann derselbe selbst nach Umgehen der Authentifizierung oder bei "Lauschangriffen" keine Daten lesen [Momut 2000]. Die unbefugte Ausführung von Applikationsfunktionen hingegen kann nur in Kombination mit Authentifizierungsmaßnahmen verhindert werden.

  Der Grad der Schutzwirkung der Verschlüsselung hängt von der Stärke des Verschlüsselungsalgorithmus ab. Zur Zeit sind Schlüssel mit einer Länge von 128 Bit aktuell. Verschlüsselung erfordert je nach Typ ein gewisses Maß an Prozessorressourcen und verstärkt die Netzlast in geringem Ausmaß.

  Moderne Verschlüsselungsverfahren lassen sich in drei Kategorien einteilen [Momut 2000].

  - **Symmetrische Verschlüsselungsverfahren**

    Dieses Verfahren funktioniert mit einem einzigen Schlüssel, der gleichzeitig zur Verschlüsselung und Entschlüsselung dient. Jedesmal, wenn der Schlüssel auf der Information angewendet wird, wird die Information entweder verschlüsselt oder wieder entschlüsselt. Wenn das Verfahren für Netzsicherheit (d.h. sichere Übertragung zwischen zwei oder mehreren Kommunikationsendpunkten) angewendet wird, muss zuvor für eine sichere Verteilung des Schlüssels gesorgt werden. Dies kann z.B. durch persönliche Übergabe (z.B. bei Vertragsabschluss) oder über den Postweg geschehen.

Symmetrische Verschlüsselungsalgorithmen haben den Vorteil, sehr schnell zu sein und wenig Prozessorleistung zu benötigen. Der Nachteil des Verfahrens ist, dass ASP und Kunde für eine sichere Übergabe des vertraulichen Schlüssels sorgen müssen.

Symmetrische Verschlüsselungsverfahren sind z.B. DES, 3DES ("Triple-DES"), RC5 und IDEA. RC5 wird z.B. von der Netzsicherheitstechnologie "SSL" eingesetzt, die von nahezu allen websprachenbasierten oder terminalbasierten ASP Basistechnologien verwendet wird.

- **Asymmetrische Verschlüsselungsverfahren (Public Key Verschlüsselung)**

Bei Public Key Verschlüsselungsverfahren besitzt jeder Kommunikationspartner einen eindeutigen privaten Schlüssel und einen korrespondierenden öffentlichen Schlüssel. Jeder Absender kann den öffentlichen Schlüssel des Empfängers benutzen, um Information zu verschlüsseln und an den Empfänger zu senden. Nur der Empfänger kann mit seinem privaten Schlüssel die Nachricht korrekt entschlüsseln. Die Schlüssel sind dabei dermaßen gestaltet, dass eine Generierung des öffentlichen Schlüssels aus dem privaten Schlüssel möglich ist, umgekehrt jedoch nicht (bzw. nicht in absehbarer Zeit).

Das Public Key-System kann auch dazu verwendet werden, um **digitale Unterschriften** zu erzeugen. Eine digitale Unterschrift ("Signatur") ist eine verschlüsselte Prüfsumme ("Hash") einer beliebigen unverschlüsselten oder verschlüsselten Nachricht, die an die jeweilige Nachricht angehängt wird. Der Empfänger entschlüsselt die Prüfsumme und überprüft mit ihrer Hilfe die Integrität der Nachricht. Gleichzeitig wird die Identität des Absenders verifiziert, denn die Prüfsumme kann nur mithilfe des zum öffentlichen Schlüssels korrespondierenden privaten Schlüssel korrekt dekodiert werden. Die folgende Grafik zeigt den Sachverhalt:

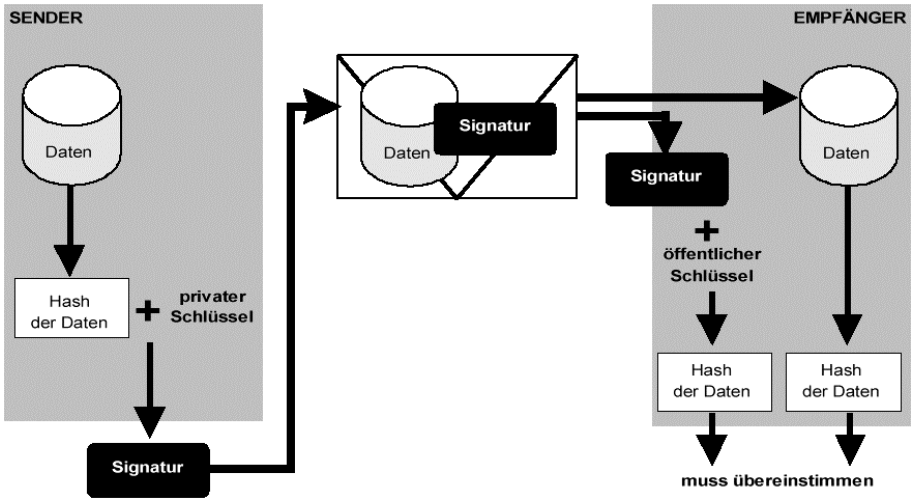

Abbildung 2-9 : Funktionsweise der digitalen Unterschrift [Gschwend 2001]

Public Key-Kryptosysteme haben den Nachteil, sehr viel CPU-Ressourcen zu verbrauchen, und werden daher typischerweise für kurze Nachrichten mit hohem Vertraulichkeitsgrad verwendet. Public Key-Verschlüsselung wird zunehmend in Hardware abgebildet, um die Performance zu erhöhen.

Zu den asymmetrischen Verschlüsselungsverfahren zählen RSA, D-H und ECDH.

- **Sicherer Schlüsselaustausch**

  Dieses Verfahren stellt quasi eine Kombination der beiden letztgenannten Verfahren dar.

  Mithilfe asymmetrischer Verschlüsselungsverfahren wird vor der ersten Nachricht ein Schlüssel eines symmetrischen Verschlüsselungsverfahrens vertraulich kommuniziert. Alle darauf folgenden Informationen werden anschließend via diesen symmetrischen Schlüssel ver- bzw. entschlüsselt. Der Vorteil in diesem Verfahren liegt in der Kombination aus der hohen Sicherheit bei der Übergabe des Schlüssels und einer hohen Performance bei der Datenver- und entschlüsselung.

- **Public Key Infrastructure (PKI)**

  Eine PKI dient der Handhabung bzw. Verwaltung von öffentlichen Schlüsseln, die zur Datenverschlüsselung, für digitale Unterschriften und Zertifikate benötigt werden.
  Mit Hilfe von **Zertifikaten** geschieht die Verbreitung von öffentlichen Schlüsseln. Die Zertifikate bestehen aus dem zu zertifizierenden öffentlichen Schlüssel, aus Angaben zum Inhaber und aus einer digitalen Unterschrift auf dieser Kombination. Diese Zertifikate werden von Zertifizierungsstellen (im Englischen: "Certification Authority" (CA) oder "Trust Center") ausgestellt. Die CA stellt eine nachprüfbare Verbindung zwischen einem asymmetrischen Schlüsselpaar und einer Person bzw. Organisation her und beglaubigt diese im Rahmen des Zertifikates mit einer digitalen Unterschrift.

  Zertifikate dienen gleichzeitig zur Integrität und zum Nachweis der Authentizität von Information. Problematisch ist die Speicherung des privaten Schlüssels auf dem Endgerät. Einen sehr guten Schutz bietet eine Kombination mit einer Benutzername/Passwort-Authentifizierung und einer Verschlüsselung. Ein bloßes Ausspähen des Passwortes reicht dann nicht mehr, um durch Maskierung unbefugt in ein ASP System einzudringen.

  Auch ASPs oder ASP Basistechnologie-Hersteller authentifizieren sich mittels Zertifikaten vor den Kunden, und zwar meist beim Log-In oder beim Download von Client-Code. Möglicherweise unsichere Inhalte wie Java Applets und ActiveX-Controls können signiert werden, um dem Benutzer zumindest die Echtheit und den Ursprung zu garantieren. Näheres zur clientseitigen Sicherheit ist im Unterkapitel "Client Sicherheit" des Kapitels "Client Herausforderungen" zu finden.

- **Physikalische Sicherheit**

  Physikalische Sicherheit umfasst den Schutz der gesamten physikalischen Infrastruktur vor Diebstahl, Unfall, Vandalismus, Feuer, Wasser und Naturkatastrophen (z.B. Erdbeben, Flut). Dazu gehört eine sichere Gestaltung der Räumlichkeiten, des Data Centers, eine strikte Zugangskontrolle sowie eine ganztägige Überwachung. Die Überwachung wird durch technische Alarm-Systeme wie Brandmelder, Wassermelder, Bewegungsmelder und Kameras, die auch regelmäßig gewartet werden müssen, gesichert. Bei den Zutrittskontrollen können technische Maßnahmen wie z.B. Identitätskarten oder organisatorische Maßnahmen wie z.B. Personenüberprüfungen zum Einsatz kommen.

2.6.5.4. Netzwerksicherheit

Die Technologien, Mechanismen und Produkte für die Netzwerksicherheit sollen eine sichere Kommunikation zwischen den Kommunikationspartnern (ASP, ASP-Partner und Kunden) herstellen. Sie bieten Authentifizierung, Zugriffskontrolle, Vertraulichkeit, Integrität, "Intrusion Detection" und Verschlüsselung. Dies alles kann jedoch nur unter der Voraussetzung der

Anwendung der Sicherheitskonzepte und -maßnahmen von Kapitel 2.6.5.3. gelingen. Größte Bedeutung kommt hier dem Netzmanagement zu.

- **Netzmanagement**

  Der Entwurf einer sicheren und durchdachten Netzarchitektur sowohl auf Seite des LANs des ASPs als auch zwischen Kunde und ASP (VPN) und die korrekte Konfiguration aller Netzwerkkomponenten stellen die wichtigsten Aufgaben des Netzmanagements dar. Da die Anforderungen und Technologien in einer ASP Umgebung sehr komplex sind, werden hier auch die meisten Fehler gemacht. Das Netzmanagement umfasst weiters umfangreiche Qualitätskontrollen und Testvorgänge.

  Zu den "best-practices" der Netzkonfiguration zählen:

  - **Nur unbedingt notwendige Protokolle und Dienste aktivieren**

    Für ASP notwendige Protokolle sind HTTP, ICA, RDP oder AIP, die durch Technologien wie SSL, IPSec oder VPN gesichert werden müssen. Alle anderen für Kunden überflüssige Protokolle und Dienste wie FTP, Telnet, Finger, whois, SNMP etc. sollten deaktiviert werden, um die Angriffsmöglichkeiten einzuschränken.

  - **Netzzugriff einschränken**

    Eine Firewall schränkt die Zugriffsmöglichkeiten ein und kann Angriffe erkennen und abwehren. DNS Server müssen sicher konfiguriert und betrieben werden.

  - **Tools zur Ausforschung der Netzstruktur wenn möglich blockieren**

    Hacker setzen Tools wie z.B. Ping, Trace Route und Port Scanner zur Ausforschung der Netztopologie ein. Ausgefeilte Tools wie z.B. das Security Analysis Tool for Auditing (SATAN) können automatisch ein gesamtes Subnetz nach möglichen Sicherheitslücken absuchen [Momut 2000]. Die Abwehr von durch Tools unterstützter Spionage ist relativ schwierig zu bewerkstelligen.

- **Firewall**

  Firewalls sind die meist benützte Form des Schutzes vor externen Hacker Attacken. Eine Firewall ist eine Netzwerkkomponente, die nur autorisierte und für den ordnungsgemäßen Betrieb vorgesehene Datenpakete auf die jeweils andere Seite passieren lässt. Sie bildet quasi einen Schutzschild zwischen den internen Netzwerken des ASPs und externen Netzwerken wie z.B. dem Internet. Das setzt natürlich voraus, dass Firewalls selbst immun gegen Attacken sein müssen [Momut 2000].

  Für ein robustes Data Center, das neben dem für die Kunden vorgesehenem Netz noch ein Intranet für ASP-Mitarbeiter beinhaltet, wird meist ein zweistufiges Firewall-System eingesetzt (eine äußere und eine innere Firewall). Zwischen beiden Firewalls befindet sich eine sog. **demilitarisierte Zone** (DMZ). Auf diese Weise ist das Intranet noch geschützt, selbst, wenn die äußere Firewall überwunden worden sein sollte. Außerdem können in dieser DMZ Rechner bzw. Server aufgestellt werden, die einer ausgewählten Öffentlichkeit (z.B. Kunden) zugänglich gemacht werden sollen, ohne dass diese gleich Zugriff auf das gesamte interne Netzwerk benötigen.

  Der ASP muss seine Firewall so konfigurieren, dass Kunden nur auf ihre eigenen Daten und Applikationen zugreifen können, andere Personen aber nicht [Rupp 2002]. Eine

wesentliche Einrichtung von jeder Firewall sind sog. **Paketfilter**. Paketfilter blockieren Pakete basierend auf ihrem Protokoll, ihrer Adresse und/oder ihrer Portnummer [iStart 2002]. Firewalls können weiters Network Address Translation (NAT), eine PKI-Infrastruktur zur Authentifizierung oder ein sog. **Intrusion Detection System** (IDS) integrieren. IDSs bieten auf Basis von mehr oder weniger intelligenten Verfahren eine Früherkennung oder nachträgliche Erkennung von Netzangriffen auf einer höheren Ebene als der Paket-Schicht und lösen bei Eintreten Alarme aus. Diese Produkte sind noch relativ neu und noch nicht ausgereift. Daher bleiben oftmals viele Angriffe unentdeckt oder es werden falsche Positivmeldungen generiert [Hollander 2002]. IDS-Systeme können keine unbekannten Attacken abwehren. Die Verwaltung der IDS-Einheit (Konfiguration, Update von Attacken-Schemas) sollte als Verpflichtung im Sicherheitsplan des ASP enthalten sein [Chiu 2001].

- **SSL (Secure Sockets Layer)**

Secure Sockets Layer (SSL) ist ein Protokoll zum Schutz eines Kommunikationskanals, das Geheimhaltung (durch Verschlüsselung), Authentifizierung und Integrität vereint. SSL ist eine standardisierte Implementierung von Public Key Verschlüsselung und digitaler Signatur, die zusammen eine PKI formen. SSL arbeitet auf Anwendungsebene. Dadurch ist es möglich, jedes beliebige Internetprotokoll durch SSL über TCP/IP zu "tunneln". Dazu wird der verschlüsselte Datenstrom einfach über einen speziellen TCP-Port geführt. Dies kann auf Anwendungsebene geschehen oder durch separate, spezielle Proxy-Server, die als Protokoll-Umsetzer dienen. Der Vorteil ist, dass SSL für nahezu alle Plattformen verfügbar ist, für Webserver, Terminalserver, Browser und Terminalclients. Der Nachteil ist, dass bei SSL die IP-Schicht (z.B. IP-Adressen) nicht verschlüsselt wird.

SSL ist ein sehr verbreitetes, flexibles und anerkanntes Verfahren. SSL kann auch zum Aufbau eines Virtual Private Networks (VPN) benutzt werden.

- **IPSec (Internet Protocol Security)**

IP-basierte Kommunikation ist grundsätzlich nicht sicher. IP bietet keinen Schutz bzw. keine Sicherheit vor folgenden Angriffen [BinTec 2001].

- Gefälschte Absenderadressen
- Inhalt von IP-Paketen verändern
- Replay-Angriffe (alte Pakete werden wiederholt gesendet)
- Unbefugtes Mitlesen von IP-Paketen während der Übertragung
- Paketumleitung an einen nicht autorisierten Empfänger
- Modifizierung des Inhaltes während der Übermittlung

Bei IPSec handelt es sich um eine Reihe von Internet Engineering Task Force (IETF) Standards, die Mechanismen zur Authentifizierung, Verschlüsselung und Integrität von IP-Paketen spezifizieren [Schnabel 2002]. IPsec wurde als integraler Bestandteil von IPv6 entwickelt. Dabei wurde aber sichergestellt, das die IPsec-Verfahren und Protokolle auch mit IPv4 nutzbar sind. Darüber hinaus kann die IPSec Implementierung nahtlos in eine Public Key Umgebung (PKI) integriert werden.

Die IPsec-Architektur besitzt die folgenden zentralen Elemente:

- **Authentification Header - Protokoll (AH)**
- **Encapsulating Security Payload - Protokoll (ESP)**
- **Internet Key Exchange - Protokoll (IKE)**

Das **AH Protokoll** bietet die Authentifizierung der Datenquelle, die verbindungslose Integrität der Datenpakete und optional einen Schutz gegen Paketwiederholung. Das **ESP Protokoll** gewährleistet Datenvertraulichkeit, verbindungslose Integrität des Datenpakets, Vertraulichkeit des Datenverkehrsflusses, Authentifizierung der Datenquelle und einen Schutz gegen Paketwiederholung. Die AH und ESP Protokolle arbeiten mit zwei verschiedenen Betriebsarten, dem Tunnelmodus für unsichere Verbindungsstrecken und dem Transportmodus für Kommunikation innerhalb desselben Netzes.

Im **Tunnelmodus** wird das komplette IP-Paket vor der Übertragung verschlüsselt. Vor den IP-Kopf wird der IPsec-Kopf und ein neuer IP-Kopf gesetzt, der nur die Daten für das Ziel-Gateway enthält. Dieses Verfahren hat drei Vorteile, nämlich dass IPSec nur in den Gateways verschiedener Netzwerke implementiert sein muss, dass als unverschlüsselte Daten lediglich die IP-Adressen des Gateway der jeweiligen Netze vorkommen (Gateway des ASP und Gateway des Kunden bzw. des ISP des Kunden) und dass auch andere Protokolle als TCP/IP über IPSec "getunnelt" werden können. Möglich ist auch eine "Tunnelung" bis zum Endgerät des Kunden, dann geht jedoch auf Kundenseite der Vorteil der Vertraulichkeit der Endgeräte-IP-Adresse verloren. Der Nachteil des Tunnelmodus ist ein gewisser Overhead an Steuerungsdaten.

Im **Transportmodus** wird nur der Datenteil des IP-Paketes verschlüsselt. Der originale IP-Kopf bleibt erhalten. Ihm wird nur ein IPsec-Kopf hinzugefügt. Dieser Modus wird innerhalb des kundeneigenen Netzes und innerhalb des ASP-eigenen Netzes benutzt. In jeder Netzkomponente, die IPSec-Pakete im Transportmodus verarbeiten soll, muss die IPSec Funktionalität implementiert sein. Das Verfahren bietet weniger Schutz, da die IP-Adressen von Client und Server unverschlüsselt bleiben. Der einzige Vorteil gegenüber dem Tunnelmodus ist der geringere Overhead an Steuerungsdaten.

Zur Schlüsselverwaltung gibt es zwei Wege, um die Verwaltung und Verteilung der Schlüssel innerhalb eines VPN durchzuführen. Neben der reinen manuellen Schlüsselverwaltung, kann auch das **IKE Protokoll** eingesetzt werden, das den Austausch von für eine gesicherte Verbindung notwendiger Parameter zwischen zwei Stationen durchführt:

- Art der gesicherten Übertragung (Authentifizierung oder Verschlüsselung)
- Verschlüsselungsalgorithmus
- Schlüssel
- Dauer der Gültigkeit der Schlüssel

Alle diese Parameter werden in der Security Association (SA) beschrieben. Jede gesicherte Verbindung bedarf einer solchen SA für jedes genutzte IPsec-Protokoll an jedem Ende einer logischen Verbindung.

Die Protokolle, die IPSec verwendet, sind von Anfang an so gestaltet worden, dass sie algorithmenunabhängig sind. Das Angebot an Verschlüsselungsalgorithmen, die zu verwenden sind, ist in einer sogenannten Security Policy Database (SPD) genau angegeben. Beispiele kryptographischer Algorithmen sind DES, 3DES, CAST und Blowfish für symmetrische Verschlüsselungen sowie SA und DSA für asymmetrische Verschlüsselungen. Eine standardisierte Reihe von Algorithmen ist durch die IETF spezifiziert, um herstellerübergreifende Interoperabilität zu gewährleisten.

IPSec ist noch relativ neu und benötigt zum Betrieb auf den dezidierten IPSec-Endpunkten wie Client, Server, Gateways oder auch Router spezielle Treiber, die erst noch einen entsprechenden Verbreitungsgrad erreichen müssen. Diese Treiber sollten idealerweise in Hardware realisiert sein, da die kryptographische Echtzeitverarbeitung sehr rechenaufwendig ist.

IPSec gewinnt im ASP-Bereich stark an Bedeutung, da damit noch sicherere VPNs als mit SSL aufgebaut werden können.

- **VPN (Virtual Private Network)**

VPN ist ein sehr allgemeiner Begriff für eine sichere, "private" Kommunikation über ein gemeinsam benutztes oder sogar öffentliches Netz (wie z.B. das Internet). Der Begriff "privat" beschreibt eine geschlossene Benutzergruppe, ähnlich die, wie sie in einem Intranet vorherrscht, samt dem erforderlichen Schutz- und Sicherheitsbedürfnis. Ein VPN gilt als ein solches, wenn es Authentifizierung, Vertraulichkeit und Integrität in einem gemeinsamen (eventuell öffentlichen) Netz bietet. VPNs sind sowohl für gemeinsam benutzte private Netze wie DSL, ATM und Frame Relay verfügbar, um die Vertraulichkeit auf einen noch engeren Benutzerkreis einzuschließen, als auch für öffentliche Netze wie z.B. das Internet. Dabei operieren sie auf OSI-Schicht 2 oder OSI-Schicht 3 [Bird 2001]. Optional bieten manche (wenige) VPNs auch Qualities of Service an. Bei VPN-Angeboten für das Internet ist dies jedoch eher die Ausnahme als die Regel.

VPNs erweitern "Ende-zu-Ende"-Kommunikationssicherheitstechnologien wie SSL und IPSec um Verwaltungstechnologien für eine größere Benutzergruppe (bis zu 1000 Knoten) zu kompletten VPN-Lösungen. Je nach Sicherheitspolitik kann der ASP für jeden einzelnen Benutzer ein VPN einrichten oder für Benutzergruppen (wenn der ASP-Kunde z.B eine Firma mit mehreren Mitarbeitern ist). Der Mehrwert von VPNs zeichnet sich meist durch die Transparenz der Kommunikationssicherheit aus. So wird jede Kommunikation innerhalb des VPNs automatisch über den Sicherheitskanal geführt ("getunnelt") und man ist nicht auf die Nutzung von bestimmten Protokollen (z.B: HTTP) oder Applikationen (z.B. Terminalclient oder Browser) beschränkt. Den Großteil eines VPNs verwaltet der ASP zentral, allerdings ist dezentral ein sogenanntes "Security Gateway" vonnöten, das eine Firewall, ein Router oder ein sonstiges VPN-Gerät sein kann. Dieses Gerät kann entweder beim Kunden stehen ("**customer premises equipment**" (CPE)) oder bei einem ISP oder VPN-Provider ("**network-based VPN**") und von diesem verwaltet werden (Outsourcing) [Bird 2001]. Die folgenden Grafiken [Corecom 2001] zeigen diese zwei unterschiedlichen VPN-Architekturen:

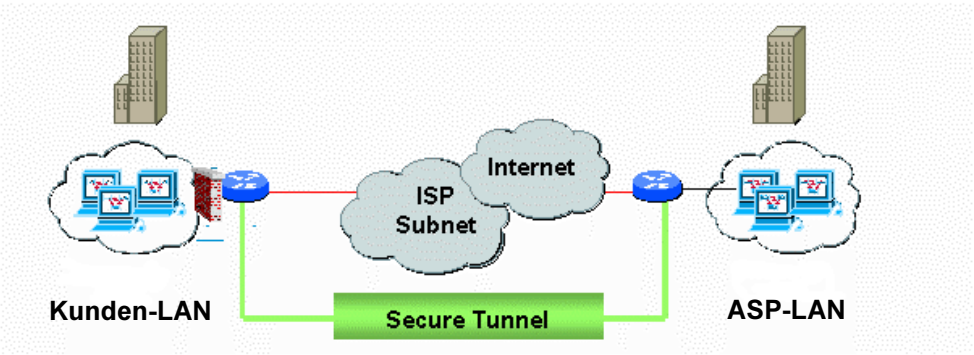

Abbildung 2-10 : Site-to-Site VPN mit CPE [Corecom 2001]

VPN mit CPE ist besonders dann zu empfehlen, wenn ein internes LAN beim ASP-Kunden schon existiert. Das interne LAN (Ethernet, ATM, Frame Relay) kann entweder gesichert oder ungesichert sein, die VPN-Funktionalität endet im allgemeinen beim Gateway des Kunden. Es gibt allerdings auch die Möglichkeit der sicheren Übertragung bis zum Endgerät ("End-to-Site"-Variante), dann müssen jedoch auch auf den Endgeräten VPN-Komponenten installiert sein (siehe "Transport Modus" unter "IPSec").

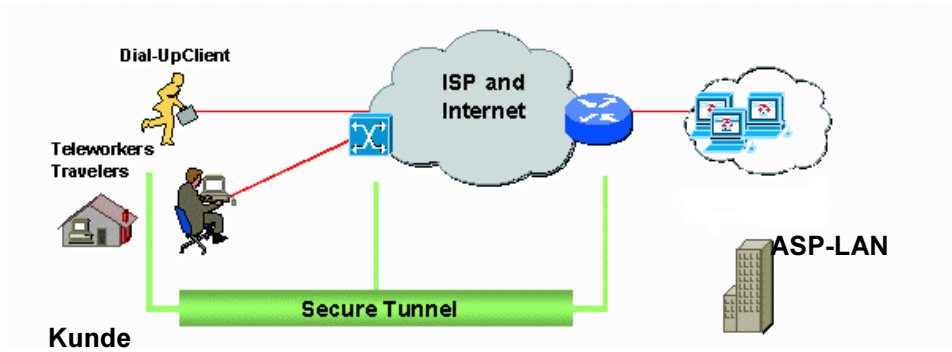

Abbildung 2-11 : Network-based VPN [Corecom 2001]

Bei Außenstellen, Heimarbeit sowie kleinen und mittleren Unternehmen ohne IT-Abteilung sind Dial-In-, Kabel- und DSL-Verbindungen die einfachere und billigere Lösung. Dann kann die gesamte VPN-Verwaltung abseits des Kundennetzes geschehen. Der in der Grafik gezeigte "Secure-Tunnel" zwischen Kunde und Einwahlknoten ist optional. Für die Eingliederung der "Last-Mile" in das VPN gilt das unter "VPN mit CPE" gesagte, allerdings ist die Netzsicherheit zwischen Kunde und Einwahlknoten kritischer zu betrachten.

VPNs sind in Zusammenhang mit ASP sehr sinnvoll, da hierbei die Netzsicherheit vollständig einem Service Provider (ASP, ISP oder VPN-Provider) überlassen werden kann.

2.6.5.5. Server- und Applikationssicherheit

Ebenfalls eine große Herausforderung ist es, die Unverwundbarkeit von Software herzustellen. In einer ASP-Umgebung sind die zentralen Softwarekomponenten, grob gesprochen, das Serverbetriebsystem, die Server-Software (Web Server, Terminalserver) und die Applikationen, die es vor unbefugter Veränderung, Benützung, Vervielfältigung und generell vor unberechtigtem Zugriff zu schützen gilt. Als Nutzer von Applikationsdiensten wünschen die Kunden, dass die ASP-Anwendungen sicher und vertrauenswürdig sind. Sie sollen die von ihnen verlangte Funktionalität, und nicht etwa Viren oder sog. "trojanische Pferde" beinhalten [Kehrer 2000]. Da Software-Sicherheit sehr stark mit Verfügbarkeit von Funktionalitäten korreliert, die u.U. unternehmenskritisch für die ASP-Kunden sind, liegt in der Softwaresicherheit eine der größten Aufgaben eines ASP. Hinzu kommt, dass das Wesen des Application Service Providings darin liegt, dass eine Applikation von mehreren Benutzern gleichzeitig verwendet werden kann. Die Folgen von Mängeln in der Softwaresicherheit machen sich dann bei mehreren ASP-Kunden gleichzeitig bemerkbar.

Software wird generell immer komplexer, funktioneller und wertvoller. Dies verlangt immer ausgefeiltere Sicherheitskonzepte für Software. Das größte Problem jedoch stellen Software-Fehler, sog. "**Bugs**", dar.

- **Softwaremanagement**

  Die Aufgaben des Softwaremanagements sind Server-, Betriebsystem-, Applikations- und Datenbanksoftware nach den Anforderungen der Security Policy sicher zu gestalten.

  Dies geschieht durch geregelte Qualitätskontrollen, durch Code-Reviews, durch Testen, durch Benutzer-Protokollierung, durch Zugriffskontrolle und Verwaltung der legitimierten Benutzer sowie durch Virenschutz.

  Zum Softwaremanagement gehört auch eine Konfigurierung der Software durch ein hochqualifiziertes Personal. Die Default-Konfiguration von Web Servern sollte zum Beispiel auf keinen Fall beibehalten werden. Zu einer korrekten Web Server- bzw.

Terminalserver-Konfiguration und einer sicheren Softwareimplementierung gehören folgende "best-practices" [Chiu 2001]:

- Pufferüberläufe wenn möglich verhindern
- Interpreter, Shells und Konfigurationsdateien außerhalb des Serververzeichnisses ablegen
- Alle nicht unbedingt notwendigen Dienste, Applikationen und Benutzeraccounts entfernen
- Zugriffe auf die Eingabeaufforderung oder eine Shell verhindern
- Zugriffe auf Systemfunktonen (z.B. Systemsteuerung) verhindern
- Applikationsfunktionen mit hohem Ressourcenverbrauch deaktivieren
- Benutzereingaben immer verifizieren

- **Software-Fehler (Bugs)**

Software-Fehler sind besonders gefährlich, da sie meist über einen längeren Zeitraum vom ASP oder ISV unentdeckt bleiben. Bugs können sich in jeder Software befinden, in der Firmware (BIOS), im Betriebsystem, in der Server-Software, in den Skript-Interpretern und in den Applikationen. Bugs öffnen grundsätzlich für jede Art von Angriff Tür und Tor, können aber auch durch korrekte Bedienung aktiviert werden. Da Applikationen gemeinsam benutzt werden, akkumulieren sich die Folgen eines Bugs auf alle Kunden, die diese Applikationen benutzen. Gegen Bugs schützt keine der vorgestellten Sicherheitsmaßnahmen (da die Implementierungen der Sicherheitsmaßnahmen selbst wieder Bugs enthalten können), sondern nur ein ausgezeichnetes Qualitätsmanagement und ein kontinuierliches Code-Review.

- **ACL (Access Control List)**

Eine sehr wichtige Aufgabe, insbesondere bei ASP, ist die Trennung der individuellen Kundendaten bei gemeinsamer Nutzung von Medien und Ressourcen. Der ASP betreut Daten von u.U. konkurrierenden Unternehmen, die auf keinem Fall dazu befähigt sein dürfen, fremde Daten auszuspähen.

Für jeden Kunden müssen Zugriffs-Schemas definiert sein. Ein Zugriffs-Schema enthält Information, wer welche Daten wie verwenden darf, und wird meist in einem Verzeichnisdienst (z.B. LDAP) abgebildet. Manche Systeme auf Websprachenbasis (z.B: IBM WebSphere) bieten sogar Zugriffskontrolle auf Ebene von Applikationsmodulen.

- **Virenschutz**

Malware ist der korrekte Oberbegriff für Viren, Würmer und trojanische Pferde, wird im folgenden jedoch wie üblich zu "Viren" zusammengefasst. Auf die Begriffsbestimmung und die Funktionsweise von Viren wird hier nicht näher eingegangen, siehe dazu [Luckhardt 2002].

Durch die Client-/Serverarchitektur ist ASP prinzipiell genauso für Virenattacken gefährdet wie jedes andere vernetzte Computersystem. Der Unterschied ist, dass sich nun die Sicherheitsvorkehrungen an einem zentralen Punkt, dem ASP, konzentrieren und daher bei Virenattacken alle Kunden eines ASPs gleichzeitig und gleich hoch gefährdet sind.

Ein vertrauenswürdiger ASP erscheint jedoch weniger anfällig gegen Virenattacken zu sein als herkömmliche vernetzte IT-Systeme. Durch die "economies of scale" kann ein ASP wesentlich aufwendigere Schutzmaßnahmen einsetzen, bei gleichzeitig niedrigem Preis für die Kunden. Sicherheitsexperten planen, implementieren und analysieren den Schutz vor

Viren. Der Druck auf den ASP, einen starken Virenschutz zu bieten, ist sehr hoch, da beim ASP vertrauliche Daten hinterlegt werden. Da i.A. nur einmalig zertifizierter Client-Code (Browser, Terminalclient) installiert werden muss, ist das Risiko der Infizierung des Client-Rechners sehr gemindert.

Ein guter Virenschutz beginnt mit einer strengen Sicherheitspolitik und einem effektiven Intrusion Detection System, das die Einschleusung von Viren von vornherein verhindert. Würmer sollten schon beim Eintreffen über das Netz abgefangen werden. Um in das Data Center einzudringen, machen sich Viren oft Fehlplanungen oder mangelnde Konfiguration der Sicherheitsarchitektur oder Software-Fehler zunutze. Zum Beispiel muss bei Einbinden des Clientdateisystems bei Terminalsystemen auf eine strikte Zugriffsbeschränkung geachtet werden.

Mitarbeiter schleusen Viren oft durch Leichtsinn oder Mutwillen ein. Zur Installierung von Software sollten nur speziell geschulte und vertrauenswürdige Mitarbeiter befugt sein. Einzelne Server sollten so weit als möglich isoliert werden, damit sich der Schaden bei Virenbefall in Grenzen hält. Backup-Systeme können im Notfall wertvolle Daten rekonstruieren.

Vor Viren geschützt werden müssen Dateiserver, Gateways und E-Mail-Server [Neefe 2000]. Den Großteil der Virenbekämpfung erledigt eine Anti-Viren-Software, die ständig automatisch aktualisiert werden muss.

Zu tatsächlichen Virenangriffen auf ASPs gibt es noch keine frei verfügbaren Studien.

- **Zertifikate**

Bei der Kontrolle von Software stellt sich für ASP und Kunden nicht nur die Frage der Qualität, sondern auch die Frage der Originalität und des Ursprungs. Ein ASP schreibt selten selbst die Software, die er den Kunden anbietet, sondern bezieht sie über einen ISV.

Der ISV sollte die Echtheit der Software, die er dem ASP anbietet, mittels Zertifikate (siehe Kapitel 2.6.5.3.) beweisen können. Zertifikate alleine können jedoch nicht für eine fehlerfreie Software garantieren. Ein beglaubigtes Qualitätsurteil von unabhängigen Softwaretestern gibt weitere Auskünfte über die Softwaresicherheit.

Die Verantwortungsbereiche (Kunde, ASP oder ISV) für die Folgen von fehlerhafter und unsicherer Software sind genau abzuklären. Dies gilt auch insbesondere bei Software für die Clients (Browser, Terminalclients, proprietäre Client-Software).

Zumindest ActiveX-Controls und Java Applets bieten dem Benutzer die Möglichkeit, die Echtheit und den Ursprung der Software zu verifizieren. Näheres hierzu im Kapitel "Client Sicherheit".

2.7. Client Herausforderungen

2.7.1. Benutzerinterface-Programmlogik

Beim UID (User Interface Design) und beim Entwerfen der dazugehörigen Programmlogik (bei Websprachen) sollte generell Rücksicht auf die technischen Einschränkungen von ASP genommen werden. So sollte im Hinblick auf Performance auf Grafiken (Splash-Screens, Logos, Hintergrundgrafiken) ganz oder teilweise verzichtet werden. Animierte Grafiken sollten bei einer ernsthaften Applikation ganz vermieden werden. Klänge, ebenfalls aus Performancegründen, sollten nur spärlich eingesetzt werden, und wenn, dann nur in geringen Auflösungen.

Durch die flexible Nutzung von ASP mit verschiedenen Endgeräten stellen sich beim UID (User Interface Design) neue Herausforderungen in Design und Programmierung:

- **Websprachen**

Bei Websprachen gibt es eine gebräuchliche und praktische Lösung: Der Client informiert den Server über das verwendete Betriebsystem und den verwendeten Browser. Daraufhin sendet oder generiert der Server Client-Code (HTML, WHTML, JavaScript) für die Darstellung eines geeigneten Benutzerinterface. Die Codegenerierung wird durch eine Serverprogrammlogik vorgenommen, die ebenfalls in einer speziellen Programmiersprache (Active Server Pages, PHP, CGI, Java Server Pages, Servlets) implementiert ist, und neuerdings (bei BEA Weblogic, Microsoft .NET und IBM WebSphere) von der Applikationsprogrammlogik (siehe Kapitel 2.6.1.2.) getrennt wird (Trennung von Benutzerinterface und Applikationsprogrammlogik). Durch die Formulierung in bestimmten Serversprachen (ASP (Active Server Pages), ASP.NET, JSP, Java) kann man die Verwendung von Client-Code nahezu vollständig vermeiden und erreicht dadurch eine größere Geräteunabhängigkeit und braucht bei der Programmierung keine Fallunterscheidungen zu implementieren.

Anhand des in Kapitel 2.1.2. bereits vorgestellten ASP-Produkts **Support.Oracle.com** soll nun die Benutzerinterfaceprogrammlogik von websprachenbasierten Systemen mithilfe dessen Quellcode dargestellt werden:

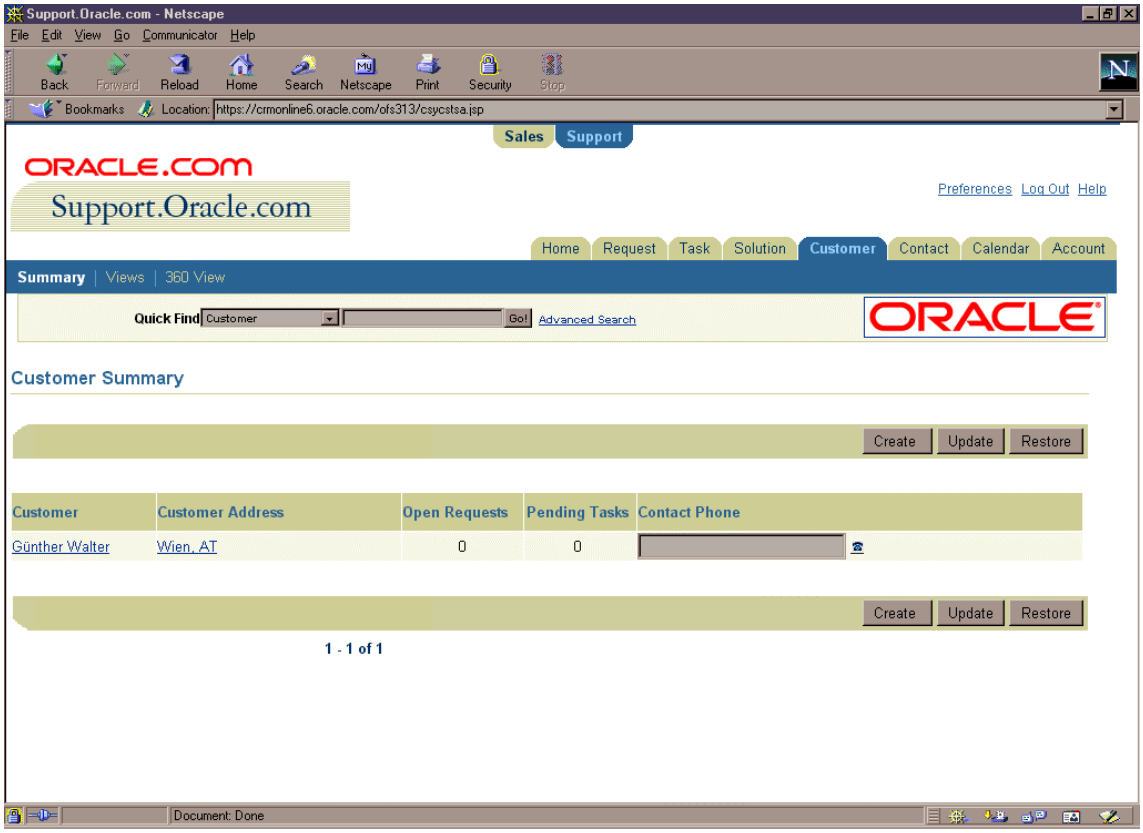

Abbildung 2-12 : Anzeige nach Aufruf des Kunden "Günther Walter" in Support.Oracle.com

Am Server werden die Seiten durch die in den JSP-Klassen (Anwendungsprogrammlogik) definierten Anweisungen dynamisch generiert, damit sich die Dynamik der Anwendungsprogrammlogik auch in der Benutzerinterfaceprogrammlogik widerspiegelt.

Die daraus generierte Benutzerinterfaceprogrammlogik am Client ist, so wie bei den meisten webbasierten ASP-Applikationen, ausschließlich aus HTML und JavaScript aufgebaut. Dies geschieht aus Gründen von Performance und Kompatibilität.

Wie bereits erwähnt, geht die Art der Implementierung in die Richtung, die Clientprogrammlogik als eine "Black-Box" zu betrachten, aus Sicht des Anwenders jedoch ist sie die Schnittstelle zur Anwendung. Sie bestimmt weitgehend die Möglichkeiten und Grenzen der Interaktivität von ASP-Anwendungen. Aus diesem Grund wird ein Teil des Beispiels näher untersucht:

Customer	Customer Address	Open Requests	Pending Tasks
Günther Walter	Wien, AT	0	0

Abbildung 2-13 : Kunde "Günther Walter" (Auszug der Anzeige)

Durch Klicken auf den Kunden können dessen Daten editiert werden. Die Programmlogik dazu ist sehr einfach (Auszug aus dem Quellcode):

```
<td class="tableDataCell" width="15%"><a
href="csycstdt.jsp?customerId=244719">Günther Walter</a> </td>
```

Die Verwendung eines HTML-Links ist der übliche Weg, eine solche Aufgabe zu lösen. Mittels einer für den Benutzer unsichtbaren ID (in diesem Fall 244719) wird die Serverprogrammlogik (`csycstdt.jsp`) angewiesen, die vollständigen Kundendaten aus der Datenbank zu lesen und ein entsprechendes Eingabeformular zum Client zu senden.
Ein weiterer Auszug:

Abbildung 2-14 : Buttons (Auszug der Anzeige)

"Create" führt auf eine Eingabemaske zum Anlegen eines neuen Kunden, "Update" aktualisiert die Suche nach dem Kunden, und "Restore" listet alle Kunden ohne Suchfilter.
Dazu die Clientlogik:

```
<input type="button" name="cancel2222" value="Create "
onClick="document.location='csycstcr.jsp'">
<input type=submit value=Update name="submit4232">
<input type=button value=Restore
onClick="location.href='csycstsa.jsp';">
```

Das Click-Ereignis des ersten Buttons führt auf eine neue Seite (`csycstcr.jsp`), das Drücken von "Update" übermittelt der aktuellen Seite die ID "submit4232", die die aktuelle Suchanfrage identifiziert und nochmals anfordert, und "Restore" lädt die aktuelle Seite erneut ohne Übergabe eines Parameters. Dies identifiziert eine Suchanfrage ohne Suchfilter. Auf diese Art werden Benutzerereignisse an den Server weitergeleitet.

Abschließend lässt sich bemerken, dass sich z.Z. die Benutzerinterfaceprogrammlogik von HTML/JavaScript-basierten ASP-Applikationen nicht wesentlich von anderen Internetseiten unterscheidet, die keine Applikationsfunktionalität integriert haben.

Webbasierte Systeme mit Java- oder Flash-Front-Ends werden wegen ihrer geringen Verbreitung im ASP-Bereich hier nicht näher untersucht, werden aber einer Untersuchung bezüglich ihres Benutzerkomforts in Kapitel 2.7.2. unterzogen.

- **Terminals**

Terminalclient-Basistechnologien haben keine Benutzerinterfaceprogrammlogik, da diese bereits in den Applikationen enthalten ist. Die Aufgaben der Einheit des Terminalclients, die sich um die Verarbeitung der Ein-/Ausgabe kümmert, sind:

- **Verarbeitung der Eingabedaten**

  Die Eingabedaten umfassen: Tastatur- und Mauseingaben, Daten von einem Scanner, einer Videokamera, aus der Zwischenablage, von lokalen Laufwerken oder einer speziell integrierten Komponente (z.B. über die Virtual-Channels von Citrix).

  Die Daten können vor ihrer Übertragung zuerst gepuffert, verschlüsselt, komprimiert oder zusätzlich lokal verarbeitet werden ("lokales Echo von Texteingaben"). Danach werden diese Roh-Daten über ein entsprechendes Protokoll zum Server gesendet (ICA, RDP, AIP).

- **Verarbeitung der Serverdaten**

  Die Serverdaten umfassen grafische Bitmap-Information, Druckerdaten und ebenfalls Daten aus der Zwischenablage, von lokalen Laufwerken oder speziell integrierten Komponenten.

  Die RDP- oder ICA-Pakete werden über einen speziellen Port von dem Client-Kontrollprogramm erkannt, empfangen und optional entschlüsselt und dekomprimiert. Je nach Bestimmung werden grafische Daten visualisiert, Druckerdaten zum Drucker gesendet, Kommandos und Dateien ans lokale Dateisystem gesendet oder Daten in der lokalen Zwischenablage abgelegt.

2.7.2. Benutzerfreundlichkeit und Anpassbarkeit

Neben Kosteneinsparungen und schnellerer Markteinführung sollen ASP Applikationen mindestens so bequem zu bedienen sein wie herkömmliche Software. Die Mechanismen, die die Benutzerfreundlichkeit von ASP Lösungen beeinflussen, laufen zwar nicht alle nur am Client ab, sie werden aber nur dort während der Benutzung wahrgenommen.

Der Grad des "Komforts" bei der Softwarenutzung ergibt sich aus vielen Einzelaspekten:

2.7.2.1. Funktionalität der Applikation

Hinsichtlich theoretisch machbarer Funktionalität gibt es bei herkömmlicher Software und ASP kaum Unterschiede. Die Funktionalität einer ASP Applikation wird aber oft eingeschränkt, damit die anderen Aspekte von ASP stärker zum Tragen kommen, und somit der Gesamtkomfort und die praktische Verwendbarkeit der Anwendung gesteigert werden kann. Denn es ist nicht die Funktionsvielfalt einer Applikation alleine, auf die der Kunde Wert legt, sondern auch die nun folgenden anderen praktischen Aspekte, die dem Kunden ein schnelleres und komfortableres Arbeiten mit der Applikation ermöglichen sollen:

2.7.2.2. Komfort des Endgerätes

Nicht immer ist der PC das ideale Endgerät für ASP. Verschiedene Hersteller (die Firma "Wyse" hat sich auf diesem Gebiet einen Namen gemacht) bieten sog. "**Thin-Clients**" an. Das sind im Prinzip "abgespeckte" PCs, ohne Diskettenlaufwerk und CD-ROM-Laufwerk, manchmal sogar ohne Festplatte, in einem platzsparenden Gehäuse untergebracht. Was diese Geräte komfortabel macht, ist das Vorhandensein des Betriebsystems (und ev. des Internetbrowsers) in einem FLASH-EPROM. Dadurch startet das Betriebsystem inkl. Log-In in weniger als 30 Sekunden (laut netConnect Graz). Die Geräte sind in der Regel vorkonfiguriert und bedürfen durch die reibungslose Zusammenarbeit der verwendeten Komponenten kaum einer Wartung. Besonders komfortabel ist die Möglichkeit, das Gerät jederzeit komplett abschalten zu können. Beim Wiedereinschalten kann man in wenigen Sekunden wieder dort weiterarbeiten, wo man unterbrochen hat. Durch das Abschalten der Geräte in den Arbeitspausen (z.B. in der Mittagspause) können Unternehmen außerdem noch 30-40% an Strom sparen [Paiha 2001]. Auch das Fehlen eines Lüfters macht sich durch die Geräuschlosigkeit der Geräte sehr angenehm bemerkbar. Manche ASPs vermieten diese Geräte zur Applikation gleich mit dazu und schnüren daraus branchenspezifische Angebote. Zielgruppe sind vor allem Unternehmen mit starker Filialstruktur.

Auf der anderen Seite können solche minimalistischen Systeme in Bezug auf Integration (siehe Kapitel 2.4.2.) und Erweiterbarkeit eingeschränkt sein. Dies ist aber von Anwendung zu Anwendung verschieden zu bewerten, und hängt letztlich auch davon ab, für welchen Einsatz außer ASP das Clientgerät noch bestimmt sein soll.

Dann existieren noch die **mobilen Endgeräte**: Handys und Palmtops (PDAs-Personal Digital Assistant). (Laptops werden hier nicht explizit genannt, da sich ihre Bedienbarkeit kaum von einem PC unterschiedet).

Palmtops beginnen langsam an Leistungsfähigkeit, und damit auch an Komfort, zu gewinnen. Es wird daher bereits versucht, gewöhnliche Desktop Applikationen via Terminal mit diesen Geräten zu benutzen. Ein wesentliches Kriterium in diesem Zusammenhang ist die **Bildschirmgröße** (wobei sowohl die Auflösung als auch die Bildschirmdiagonale eine Rolle spielen). Die meisten MS Windows Applikationen setzen eine Bildschirmauflösung von mindestens VGA (640x480) Pixel voraus. Manche Applikationen, darunter MS Access, verweigern sogar den Start bei geringeren Auflösungen. Selbst wenn die Applikation unter solch geringen Auflösungen startet, kann die Brauchbarkeit dieser Applikation darunter leiden - die Benutzer können nicht genug Daten sehen, um vernünftig arbeiten zu können. PDAs, auf denen WinCE als Betriebsystem läuft, wie der Compaq iPAQ, haben typischerweise nahezu quadratische Bildschirme (240x320 Pixel). Die Abbildung 2-15 zeigt auf der linken Seite das Gerät während einer Microsoft Outlook Session über ICA. Rechts eine ICA Session, die auf 360x480 Pixel läuft und mittels Citrix' proprietärer Skalierungstechnologie auf 240x320 Pixel herunterskaliert wurde. Der Text wird kleiner und unschärfer, aber der Benutzer bekommt eine größere Arbeitsfläche. Drucktechnisch bedingt ist die hier gezeigte Abbildung in Bezug auf die Bildqualität und Lesbarkeit einer skalierten Citrix-Session nicht repräsentativ:

Abbildung 2-15 : MS Outlook über Citrix auf dem Compaq iPAQ

Es erscheint aber einsichtig, dass mit einem PDA derzeit Komforteinbussen in Kauf genommen werden müssen, doch ist es schwierig zu beurteilen, ob die eigentlichen technischen Unzulänglichkeiten tatsächlich nur beim Endgerät zu suchen sind: Dem ASP Gedanken "Anytime, anywhere, **on any device**" wird in der Praxis noch nicht Rechnung getragen. Somit wäre also auch zu argumentieren, dass die Mankos im Benutzerkomfort bei der Nutzung von ASP über PDAs auch in der Funktionalität der Applikation, im Komfort des Benutzerinterface und dessen Anpassbarkeit liegen.

Bei Handys ist das Angebot an Applikationsdiensten in der Minderzahl, dies nicht zuletzt wegen des unzureichenden Komfort: Keine Schreibmaschinentastatur, zu kleines Display mit geringer Auflösung, umständliche Menünavigation, da ein Gerät zur direkten Befehlseingabe (Maus oder Stift) fehlt. Hierbei können derzeit in keinem Fall dieselben Applikationen wie sie auf dem Desktop-Client benutzt werden, zum Einsatz kommen. Zur Zeit liegt es an den ASPs und Softwareherstellern, trotz der technischen Einschränkungen komfortable ASP Lösungen für Handys anzubieten. Der Wunsch der Kunden ist weniger auf dem Handy dasselbe Programm wie auf dem Schreibtischrechner zu benutzen, sondern auf denselben Daten operieren zu können. Diesem Wunsch versuchen viele ASPs durch ihr Lösungsportfolio gerecht zu werden.

2.7.2.3. Komfort des Benutzerinterface

Terminalbasierte Dienste bieten dasselbe User-Interface, wie es auch am Server läuft. Es sind z.B. bei Citrix das User-Interface von Windows NT, Windows 2000 oder X-Windows von UNIX. SCO Tarantella läuft auch auf Unix, bei diesem Betriebsystem hat man bekanntlich die Möglichkeit, mehrere verschiedene Benutzerinterfaces zu nutzen. Die Funktionalität des GUIs bei diesen Systemen unterscheidet sich nicht von einem gleichwertigen lokalen Betriebsystem. Scheinbare Komforteinbußen beim Benutzerinterface sind entweder auf eingeschränkten

Funktionalität der Applikation (siehe Punkt eins), eingeschränkter Komfort des Endgerätes (letzter Punkt) oder auf mangelnde Performance (folgender Punkt) zurückzuführen. Die von PC-Betriebsystemen gewohnt hohe Leistungsfähigkeit des Benutzerinterface räumt den terminalbasierten Systemen einen großen Vorteil gegenüber den Websprachen ein. Jedoch sollten bei Applikationen, die über Terminals betrieben werden, sofern die Benutzung mittels unterschiedlichen Endgeräten gewährleistet werden soll, die Möglichkeiten und Einschränkungen unterschiedlicher Endgeräte berücksichtigt werden. Zum Beispiel sollten die Applikationen Optionen bieten, die eine optimale Anpassung an kleine Displays von PDAs erlaubt (siehe Diskussion dieses Problems unter dem vorigen Punkt).

Bei den Websprachen sind vor allem bei HTML und JavaScript Einschränkungen zu verzeichnen: Es sind nicht alle GUI-Elemente und deren Funktionen der gebräuchlichen Benutzeroberflächen (z.B. Windows) verfügbar. Zum Beispiel gibt es keine Pull-Down-Menus und auch keine Tastatur-Shortcuts. Zum Glück lassen sich fast alle GUI-Elemente mittels Grafiken nachbilden, was sich jedoch in einer schlechteren Performance widerspiegelt. Auf http://www.gifworks.com ist ein Beispiel (eine webbasierte Bildbearbeitungssoftware) einer solchen Umsetzung zu sehen. Gerade diese Nachbildungen sind aber oftmals inkompatibel zu verschiedenen Browserversionen und Clientkonfigurationen (Bildschirmauflösung). Die Umsetzung von Applikationen, die große editierbare Arbeitsbereiche verwenden (z.B. ein Dokument in einer Textverarbeitung oder Tabellenkalkulation), ist nahezu unmöglich. Deshalb sind z.B. Textverarbeitungsprogramme auf Websprachenbasis noch nicht verfügbar.

Java-Applets sind wesentlich leistungsfähiger. Es gibt Menus und Tastaturshortcuts. Außerdem ist eine komplette Grafikengine eingebaut. Beim Umgang mit großen Arbeitsflächen stößt jedoch auch Java auf seine Grenzen. Es ist auch hier erforderlich, die gesamte Information vorab vom Server zu laden.

Java Technologie als Benutzerinterface setzt sich leider im ASP Bereich kaum durch, was anhand der verfügbaren ASP Anwendungen auf http://www.findapps.com leicht überprüft werden kann. Dies liegt zum Großteil an den langen Downloadzeiten und an der schwachen Performance.

2.7.2.4. Performance

Geschwindigkeit erhöht den Benutzerkomfort einer Applikation, für unternehmerisch tätige ASP-Kunden trägt rasches Arbeiten mit dem Computer aber auch zum Unternehmenserfolg bei.

Es sind generell wieder die zwei Aspekte von Performance zu unterscheiden:

- **Antwortzeit auf Eingaben des Anwenders bis zu einer sichtbaren (oder hörbaren) Reaktion durch das Benutzerinterface der Applikation bzw. des Betriebsystems**

  Generell lässt sich für komfortables Arbeiten festhalten, dass die Performance umso höher sein muss, je interaktiver die Aktionen sind. Eine Mausbewegung (z.B. Verschieben einer Grafik) muss nahezu verzögerungsfrei vonstatten gehen, da sonst eine rasche und genaue Positionierung nicht mehr möglich ist. [Stary 1996] spricht in diesem Zusammenhang von **Kontrolleingaben**. Kontrolleingaben lösen nicht direkt Applikationsfunktionen aus, sondern ziehen bloß eine Änderung der Benutzerschnittstelle nach sich, manchmal um eine Applikationsfunktion einzuleiten. Es sind Eingaben, die dem Benutzer unmittelbares *Feedback* geben. Beispiele für solche Vorgänge sind: Mausbewegung, Objektverschiebungen, Eingabemaskenaufrufe, Fensteroperationen, Menuaufrufe und manuell zu bestätigende Dateneingaben. Das Besondere an diesen Anweisungen ist, dass der Anwender von ihnen eine sehr kurze Antwortzeit erwartet: [Stary 1996], S140: "Systemreaktionen bei direkter Manipulation sollten nicht länger als eine Zehntelsekunde dauern.". Im Zuge einer Kontrolleingabe sollte die Summe der Verzögerungen bei Client,

Server und Netz diesen Wert nicht übersteigen. Bei Anweisungen an die Applikation (z.B. eine Eingabe durch "OK" bestätigen, um eine Applikationsfunktion auszulösen) werden längere Verzögerungen in Kauf genommen (siehe "Ausführungsgeschwindigkeit von Applikationsfunktionen")

Wesentlich für ein komfortables Arbeiten ist auch, dass die Antwortzeit des Benutzerinterface auf verschiedene Eingaben nicht zu stark von seinem mittleren Wert abweicht. Starke, scheinbar zufällige Abweichungen vom generellen "Arbeitstempo" des Benutzerinterface (z.B. ruckweise Bewegung bei Objektverschiebungen) werden vom Anwender als unangenehm, weil unerwartet, empfunden.

Bei **webbasierten Diensten** verlaufen Mausbewegungen verzögerungsfrei, da sie keine Serverinteraktivität zur Folge haben. Mausklicks werden zudem sofort optisch quittiert, auch wenn die gewünschte Applikationsfunktion noch nicht einmal aufgerufen ist. Dieses Feedback ist für den Benutzer äußerst wichtig. Auch Texteingaben werden fast immer lokal verarbeitet, sodass kaum mit merklichen Verzögerungen zu rechnen ist. Bei Pulldown-Menus ist zu unterscheiden: Wenn sie mit Javascript realisiert sind, kommt es, da jeder Menueintrag als Grafik realisiert sein muss, zu Verzögerungen durch den Download der Grafiken. Bei Verwendung von Java lässt sich dies verhindern. Verzögerungen durch das GUI bei webbasierten Diensten treten entweder durch zu intensive Nutzung von Grafiken auf, bedingt durch die Serverinteraktion beim Download, oder der Clientrechner und/oder der Browser sind zu langsam beim Visualisieren. Hierbei können wieder zu zahlreiche, umfangreiche Grafiken, oder, im speziellen, zu viele Tabellen bremsend wirken.

Besonders bemerkbar macht sich dieser Effekt dann bei stärkerer Interaktion: Jeder Zustandswechsel der Applikation, die eine GUI-Änderung mit sich zieht, ist mit einem Neuladen der gesamten Seite verbunden. Besonders störend ist dies z.B. wenn Eingaben häufig bestätigt ("OK"-Button) oder verworfen ("Cancel"- oder "Abbruch"-Button) werden müssen. Auch das Drücken des "Back"-Buttons des Browsers hat denselben Effekt. Hier wird leider selten dasselbe Performancegefühl wie bei lokalen Applikationen erreicht, da bei diesen kein Neuladen und kein Neuaufbau der gesamten GUI erforderlich ist. Weiters kommt es zu längeren Verzögerungen beim Abschluss von Dateneingaben: Da Dateneingaben zunächst lokal erfolgen, müssen die Daten vor dem Auslösen der jeweiligen Applikationsfunktion zuvor zum Server übertragen werden.

Ist das GUI einer webbasierten Applikation aber nicht zu komplex (wenig Grafiken, wenig geschachtelte Tables, wenige Plug-Ins, nicht zu komplexe Nutzung von Browsersprachen), dann gewährleistet es ausreichend kurze Antwortzeiten.

Bei **terminalbasierten Systemen** hingegen läuft die gesamte GUI am Server ab. Die Vorverarbeitung der Client-Eingaben beschränkt sich auf das Weiterleiten der Tastatur- und Mauseingaben an das am Server laufende User-Interface der Applikation. Jede einzelne Eingabe ruft eine Serverinteraktion und deshalb grundsätzlich Verzögerungen hervor. Aus diesem Grund sind bei diesen Systemen Kontrolleingaben problematisch. Allerdings bauen diese Systeme nicht die gesamte Seite neu auf, sondern nur diejenigen Teile, die sich gegenüber dem letzten Bild geändert haben [Citrix 2002]. Zudem gibt es Pufferfunktionen, die Eingaben bis zu einem vorgegebenen Maß lokal zwischenspeichern, bevor sie gesendet werden. Dadurch wird eine gleichmäßigere Verteilung der Verzögerungen erreicht. 1/10 Sekunde Verzögerung (inkl. Netzanforderung und -antwort) ist jedoch ein Wert, der in der Praxis selten erreicht wird. Das reicht aber nicht für komfortables Arbeiten, wenn der NSP nicht gerade ein Hochgeschwindigkeitsanbieter ist, der Verzögerungen von <0.1 Sekunden garantieren kann. Die Hersteller von terminalbasierten Systemen haben darauf reagiert: Citrix bietet optional zumindest die lokale Ausgabe von Texteingaben (Details im praktischen Teil).

Die Hauptprobleme bei permanentem oder sporadischem Performancemangel sind aber noch nicht gelöst: Menumarkierungen und Fensterverschiebungen folgen nicht sofort der

Maus, das Interagieren mit grafischen Symbolen (Buttons) erfolgt verzögert. Dass dabei die Ausführungsgeschwindigkeit solcher Vorgänge direkt mit der Netzgeschwindigkeit verknüpft ist, ist als Nachteil terminalbasierter Systeme einzustufen.

- **Ausführungsgeschwindigkeit von Applikationsfunktionen**

  Die Ausführungsgeschwindigkeit von einzelnen Applikationsfunktionen ist zum Großteil durch die Netz- und Servergeschwindigkeit bestimmt. Eine Untergrenze für Benutzerfreundlichkeit kann hier nicht gegeben werden: Sie hängt stark von der Erwartungshaltung des Anwenders, die von der Benutzung von lokaler Software geprägt ist, ab. Als störend können sich aber Schwankungen in der Ausführungsgeschwindigkeit bemerkbar machen: Während lokal installierte Software auf einem bestimmten Gerät i.A. immer annähernd gleich schnell ausgeführt wird, wird die Ausführungsgeschwindigkeit einer Applikation bei ASP von der momentanen Server- und Netzlast beeinflusst. Um dies zu verhindern, wünschen sich viele Kunden in den SLAs QoS- (Quality of Service) Parameter, die ihnen immer eine gleichbleibende Performance garantieren.

2.7.2.5. Zuverlässigkeit

Auch die Zuverlässigkeit und Robustheit des ASP-Betriebs trägt wesentlich zum komfortablen Arbeiten bei. In Kapitel 2.6.2. wurde das Thema "Ausfallssicherheit" aus technischer Sicht bereits behandelt. Der Eindruck an Zuverlässigkeit beim Endkunden ergibt sich jedoch aus der Summe von Server-, Client- und Netzzuverlässigkeit. Denn welche Komponente auch immer ganz oder teilweise ausfällt, eine Beeinträchtigung der momentanen Anwendungssitzung ergibt sich in jedem Fall. Es soll nun untersucht werden, inwieweit sich ein Applikationssausfall von herkömmlicher Software zu ASP-Software unterscheidet.

Es war im Rahmen dieser Diplomarbeit nicht möglich, die Zuverlässigkeit derzeitiger ASP-Dienste oder -Basistechnologien zu testen und zu bewerten, da eine ausreichend große Anzahl von Testfällen nicht zur Verfügung stand. Es lassen sich aber die Situationen, die beim Anwender im Falle einer ungewollten Anwendungsunterbrechung entstehen, und dessen Möglichkeiten zur Wiederherstellung des Applikationsbetriebes, erläutern.

Im folgenden sind einige Benutzer-Szenarios bei Ausfällen beschrieben:

- Wenn eine **Server- oder Netzkomponente ausfällt**, wird der Benutzer nach den Empfehlungen seines ASPs vorgehen. Bei Systemen, die Sessionzustände speichern, kann er zu einem späteren Zeitpunkt versuchen, sich durch Angabe von Benutzernamen und Passwort datenverlustfrei wieder mit der Applikationssession zu verbinden. Einige wenige ASP-Basistechnologien (die in dieser Diplomarbeit aber nicht behandelt werden) stellen clientseitig ein gewisses Maß an Applikationslogik zur Verfügung, sodass ein Weiterarbeiten ohne Netzverbindung zum ASP (off-line) in eingeschränkter Form möglich ist. Ist das nicht möglich, wird der nächste Schritt i.A. ein Anruf beim ASP sein. Kann der ASP die von ihm in den SLAs definierten Qualitätskriterien nicht erfüllen, so treten die dort vereinbarten Entschädigungsklauseln in Kraft. Ansonsten hat der Benutzer keinen Einfluss auf die Wiederaufnahme seiner Applikationssitzung.

- Wenn das **Endgerät oder der Browser** (z.B. durch einen Absturz) **ausfällt**, kann sich der Benutzer innerhalb einer bestimmten Zeit (meist etwa 15 Minuten) mit seiner Applikationssitzung wiederverbinden. Ist ihm das nicht möglich, so wird in den meisten Fällen eine automatische Sicherung mit anschließender Terminierung der Applikation vorgenommen. Bei einem Thin-Client-System geht der Reboot schnell vonstatten und auch

ein Browser ist rasch wieder gestartet. Bei webbasierten Systemen wird der Sessionzustand allerdings nicht kontinuierlich gespeichert, sodass Datenverluste bis zum letzten Speichern entstehen können [Paiha 2001].

Ein Vergleich, ob herkömmlicher Softwarebetrieb oder ASP-Betrieb zuverlässiger ist, lässt sich nicht anstellen. Dies ist von der Art der Applikation und dem ASP-Betrieb abhängig. Bei beiden Arten hat der Benutzer bei Ausfällen, die durch einen Applikationsfehler ("Bug") verursacht werden, keine unmittelbare Chance, einzugreifen. Die Behebung anderer Fehler werden von IT-Spezialisten im Haus (bei herkömmlicher Softwarenutzung) oder beim ASP vorgenommen.

2.7.2.6. Zugriff auf Soft- und Hardwareressourcen (Integration)

Die Arbeit mit einer Applikation oder einem Betriebsystem wird vielfach erst vollkommen durch Einsatz ihrer Schnittstellen zu Soft- und Hardware. Nahtlose Zusammenarbeit mehrerer Programme, Datenkonvertierung und Nutzung von Zusatz-Tools sind die Schwerpunkte im softwaretechnischen Bereich. Auch der Bedarf an Hardwareressourcen und Peripherie sollte gedeckt sein: Zugriff auf Festspeicher (auf Client- oder Serverseite), Drucker, Scanner, Joystick, Multimediageräte wie Soundkarten, Videokameras etc.

Eine wichtige Eigenschaft besonders bei ASP-Software ist die Kompatibilität zu einer breiten Anzahl von Datenformaten. Dies ist z.B. wichtig bei der Migration von Daten (von oder zu lokalem Softwarebetrieb oder einem anderen ASP) aber auch im B2B-Bereich (E-Business). Bei den Terminalsystemen ist für eine Datenkonvertierung das Mapping von ASP-fremdem Laufwerken eine wichtige Voraussetzung. Webbasierte Systeme bieten hingegen oft einen nachrichtenbasierten Datenaustausch (XML). Für beide Systeme müssen selbstverständlich entsprechende Funktionalitäten in den Applikationen vorhanden sein, die eine Datenübertragung und -konvertierung erlauben.

Bei jenen Ressourcen, die zum Applikationsbetrieb unbedingt notwendig sind, lässt sich nicht explizit von zusätzlichem Benutzerkomfort sprechen. Bei der Anbindung an Zusatz-Ressourcen liegen die terminalbasierten Systeme aber klar im Vorteil: Aus softwaretechnischer Sicht können durch das Vorhandensein eines Dateisystems auf Serverseite leicht Zusatz-Tools und Hilfsprogramme installiert und integriert werden. Der Anwender eines webbasierten Systems wird aber kaum ohne lokal installierte Tools auskommen, da es für den ASP ein wesentlich höherer Aufwand ist, Zusatztools, wie z.B. eine Notizfunktion oder einen Taschenrechner, in ein webbasiertes System zu integrieren.

Der Zugriff auf das Dateisystem gestaltet sich in jedem Fall einfacher, schneller und auf die Benutzerbedürfnisse besser anpassbar als bei den webbasierten Benutzerschnittstellen, die an Datenbanken angebunden sind. Besonders bei wachsender Anzahl an Daten wird der webbasierte Zugriff und die Verwaltung durch den Anwender rasch unübersichtlich, langsam und kompliziert.

Terminals können "von Haus aus" lokale Drucker und Scanner ansprechen, webbasierte Software kann dies nur mit erheblichem Programmieraufwand und erreicht derzeit nicht denselben Komfort.

Bei der Anbindung an Audiogeräte ("Soundkarten") sind beide Systeme gleichermaßen leistungsfähig und komfortabel.

Eindeutig im Vorteil bezüglich Verwendbarkeit sind Websprachen aber bei der Integration von speziellen Multimedia- und Netzwerkprotokollen, wie sie z.B. für Video-Streaming oder Voice-over-IP eingesetzt werden. Die terminalbasierten Systeme sind immer an ihr eigenes Netzwerkprotokoll gebunden, über das eine performante Übertragung solcher Inhalte nicht möglich ist. In Webseiten hingegen können sehr einfach Multimediainhalte (wie von Realplayer, Quicktime, etc.) integriert werden. Auf Clientseite wird lediglich ein "Plug-In"

(beim Netscape-Browser) bzw. eine "ActiveX-Control" (beim Internet-Explorer) gebraucht, um die Funktionalität des Browsers zu erweitern.

Es bleibt zu hoffen, dass auch die Terminals bald technisch dazu in der Lage sind, andere Protokolle zu integrieren, damit z.B. Multimediainhalte mit dieser Technologie komfortabel nutzbar werden.

2.7.2.7. Individuelle Anpassbarkeit

Hier ist zwischen server- und clientseitiger Anpassbarkeit zu unterscheiden: Die serverseitige Anpassbarkeit ist lediglich ein Resultat der Funktionalität einer Applikation. Es gibt keine ausgeprägten Unterschiede zwischen Websprachen und Terminals. Anzumerken ist lediglich, dass einige Benutzereinstellungen beim terminalbasierten System mittels des Serverbetriebsystems vorgenommen werden können und daher auf Applikationsebene nicht implementiert werden müssen. Ein Teil der Anpassungsmöglichkeiten ("Systemsteuerung") bleibt den Benutzern aber aufgrund von Sicherheitsauflagen leider vorenthalten. Manche dieser Einstellungen (z.B. Bildschirmauflösung) können aber mittels des lokalen Betriebsystems vorgenommen werden.

Wesentliche Unterschiede zeigen sich jedoch bei der clientseitigen Anpassbarkeit:

Bei den **Terminals** gibt es große Unterschiede zwischen den eigenständigen Terminalclients und den browserbasierten Terminalclients.

Die browserbasierten Kontrollprogramme haben meist eine relativ kleine Dateigröße (etwa 500KB) und bieten überhaupt keine Möglichkeiten zur Anpassung an die persönlichen Bedürfnisse.

Die eigenständigen Kontrollprogramme benötigen zwar eine längere Downloadzeit (3-5MB), sie bieten jedoch reichhaltige Anpassungsmöglichkeiten. Genauso wie der Browser muss die Datei ja nur bei Updates heruntergeladen werden. Die umfangreichsten Konfigurationsmöglichkeiten bietet Citrix:

Es lassen sich Anzeigemodi (Anzeige im Fenster oder gesamter Server-Desktop, Skalierung auf niedrigere Bildschirmauflösungen, Maximalanzahl an Farben, Fenstergröße), Spracheneinstellungen, Tastatureinstellungen und Tastenkombinationen, Clientgerätezuweisungen (Clientlaufwerke, Clientdrucker, COM-Anschlüsse), Audioeigenschaften, Mechanismen zur Steigerung der Performance (Bitmap-Cache, lokales Textecho, lokales Mausklickfeedback, Datenkomprimierung) sowie Verbindungs- und Verschlüsselungsoptionen konfigurieren. Viele dieser Einstellungen können auch benutzerspezifisch angepasst werden.

Bei den **Websprachen** werden die Anpassungen mit Hilfe des Browsers vorgenommen. Die Einstellungsmöglichkeiten unterscheiden sich etwas. Mit dem Browser können Anzeigemodi (Farben, Schriftarten, Bilder, Videos, Animationen, Sound, Symbolleisten), Spracheneinstellungen, Eingabehilfen (z.B. Texteingaben automatisch vervollständigen), Mechanismen zur Steigerung der Performance (Internet-Cache und Proxy) sowie Sicherheitseinstellungen (Zertifikate, Java, ActiveX, JavaScript, Cookies) konfiguriert werden. Die Audioeinstellungen werden mit der lokalen Systemsteuerung vorgenommen.

Beide Basistechnologien lassen sich auf die individuellen Bedürfnisse anpassen. Ein großer Nachteil der Browser ist allerdings, dass viele ASP-Angebote auf Websprachenbasis nicht mit individuellen Anzeigemodi kompatibel sind und zudem auf verschiedenen Browserversionen unterschiedlich angezeigt werden.

2.7.2.8. Kundenunterstützung

Die Kundenunterstützung beginnt mit Dokumentationen (in elektronischer Form oder auf Papier) und endet mit professionellen Beratungs- und Schulungsserviceleistungen. Das

Benutzerhandbuch (neben technischen Dokumenten zum ASP-Betrieb) sollte immer auch auf Papier vorhanden sein. An Onlinehilfen gibt es Such- oder Problemlösungsassistenten und Online-Benutzerhandbücher.

Erfahrungsweise sind die Onlinehilfen der herkömmlichen Software (welche über Terminals betrieben werden kann) wesentlich leistungsfähiger, komfortabler zu bedienen und performanter als die der webbasierten Anwendungen. Dies liegt zum Teil in der hohen Interaktivität und der großen Datenmengen begründet, die von Onlinehilfen gefordert werden, aber von webbasierten Systemen nicht geliefert werden können.

Wegen der starken Abhängigkeit vom Serviceprovider ist eine sehr gute Kundenunterstützung erforderlich. Der Kunde muss jederzeit über die Kontrolle seines persönlichen Benutzerprofils (Kosten, SLAs, etc. siehe Kapitel 2.3.1.) verfügen können. Dafür gibt es eigene leistungsfähige und komfortable Programme ("Customer Care"-Applikationen, meist webbasiert). Sie bieten auch eine breite Palette an Schnittstellen (E-Mail, Telefon, Fax etc.) zur Kommunikation mit Dienstleister-Mitarbeitern bei Problemen ("Call Center").

Als Teil des Operation Managements wurde die Kundenunterstützung bereits in Kapitel 2.6.4 vorgestellt.

2.7.3. Mächtigkeit des Clients

"Mächtigkeit des Clients" soll die Flexibilität, Erweiterbarkeit und Zukunftssicherheit der Clienttechnologie bezeichnen, und zwar in technischer Hinsicht. Hierauf haben die ASP Basistechnologie und insbesondere die Kommunikationsprotokolle Einfluss. Eine Clienttechnologie kann nur so leistungsfähig sein, wie das eingesetzte Kommunikationsprotokoll.

Die Terminalsysteme verwenden spezielle Protokolle, die auf Präsentations-Daten (Grafik) optimiert sind. Da Sessionzustände kontinuierlich gespeichert werden, müssen die Protokolle dafür entsprechend ausgerichtet sein. Verbindungsorientierte Protokolle im Zusammenspiel mit kontinuierlichem Bildaufbau und der damit hohen Interaktivität (ohne Seitenorientierung) sind eine gute Basis für die Zukunftssicherheit der Terminaltechnologien. Auf der anderen Seite ist es schwer vorstellbar, wie gänzlich andere Protokolle, wie sie z.B. für Voice-Over-IP oder Video eingesetzt werden, in die Terminalprotokolle integriert werden sollen, trotz der Technik der "Virtual Channels".

Außerdem ist die Client-Logik der Terminalsysteme relativ starr. Die Funktionalität des Browsers wird praktisch komplett deaktiviert. Das bedeutet, dass Terminalsysteme nicht von zukünftigen Weiterentwicklungen in der Browsertechnologie profitieren können.

Da in den GUI-Daten keine Semantik enthalten ist, ist es nur schwer möglich, die Anzeige auf die Fähigkeiten verschiedener Endgeräte anzupassen. Der Ansatz der Clienttechnologie bei den Terminals ist eher ein generischer: Die Client-Logik ist universell gestaltet und wird einmalig separat vom Applikationsdienst erworben (mittels Download). Sie hat als Native-Applikation vollen Zugriff auf Client-Ressourcen. Bei den Websprachen wird die Client-Logik hingegen in Echtzeit von der jeweiligen Applikation zur Verfügung gestellt, was ein höheres Maß an Flexibilität, aber auch ein geringeres Maß an Kompatibilität darstellt. Aufgrund der Security haben nur wenige Technologien Zugriff auf Client-Ressourcen (ActiveX und Java), und dies nur auf ausdrücklichen Wunsch (Bestätigung per Dialogbox) des Anwenders.

Auf Protokollebene erscheinen die Websprachensysteme den Terminalsystemen unterlegen. HTTP ist nicht verbindungsorientiert. Es können immer nur ganze Internetseiten per HTTP übertragen werden. Dies schränkt die Interaktivität sehr ein. Dem versucht die Clientlogik gegenzusteuern. Sie erscheint bei den Websprachen flexibler: Mit Java, JavaScript, ActiveX, Plug-Ins und Macromedia Flash gibt es eine Reihe von Technologien zur Auswahl.

Allerdings muss beachtet werden, dass leistungsfähige und komplexe Clientlogiken sehr umfangreich werden können, wodurch es bei jeder Applikationssitzung zu langen Downloadzeiten kommen kann, was von ASP-Kunden nicht akzeptiert wird.

Zudem stellen komplexe Websprachen-Clientlogiken höhere Leistungsanforderungen an das Endgerät.

Für hochinteraktive Programme erscheint HTTP nicht zukunftssicher. Es wird sich allerdings zeigen, ob HTTP das einzige Protokoll sein wird, was zukünftige Browser unterstützen werden. Ähnlich die zukünftigen Anforderungen an Terminalsysteme: Für mehr Flexibilität müssen Möglichkeiten geschaffen werden, Semantik (GUI-Objekte, Attribute etc.) und andere Protokolle integrierbar zu machen.

2.7.4. Client Sicherheit

Auf den ersten Blick erweckt die ASP-Architektur (serverseitige Installation und Datenverarbeitung, zentraler Virenschutz, Verschlüsselung etc.) den Eindruck, die Client-Sicherheit spiele nur eine untergeordnete Rolle. Tatsächlich ist dem nicht so.

Das Endgerät und die darauf laufende Software (Betriebsystem, Browser etc.) unterliegen i.A. nicht einer unmittelbaren Kontrolle durch den ASP. Schließlich betreffen die Vereinbarungen mit dem ASP nur den Schutz der Kundendaten innerhalb des ASP-Verantwortungsbereichs (Server und Netz). Dies ruft ein Verantwortungs- und Sorgebewusstsein für die Interaktion mit dem Endgerät beim Benutzer hervor.

Die Client-Security gewinnt im Speziellen dann an Bedeutung, wenn das Endgerät nicht nur zur ausschließlichen ASP-Nutzung verwendet wird, also andere als für den ASP-Betrieb notwendige Daten auf dem Clientgerät geschützt werden sollen. ASP-Kunden sind weiters in Hinblick auf Datenschutz, Firmengeheimnisse und Privatsphäre sehr darauf bedacht, dass solche Daten nicht unbeabsichtigt zum ASP oder zu anderen ASP-Partnern oder -Kunden in falsche Hände gelangen. Der Anwender erwartet sich von der Clienttechnologie, dass Zugriffe auf die Clienthard- und -software nachvollziehbar und sicher sind.

Die Gefahr, dass unbeabsichtigt nicht entsprechende Daten zum ASP gelangen könnten, ist bei ASP viel eher als beim bloßen "Surfen" gegeben, da von einer autonom arbeitenden Clientprogrammlogik aktiv Inhalte zum ASP transportiert werden. So könnte etwa eine browserbasierte Anwendung auf dem Rechner gespeicherte Daten "ausspähen", löschen oder verändern. Ob dies der ASP vorsätzlich oder unabsichtlich verursacht, ist aus Kundensicht unerheblich.

So muss erst das Kundenvertrauen in die vom Provider über das Netz zur Verfügung gestellten Daten hergestellt werden. Hier ist oft eine gewisse Skepsis gegeben, ob die übermittelten Inhalte (z.B. verwendete Protokolle, ActiveX-Terminalclient, Java, JavaScript, etc.) sicher sind und für ihre Echtheit garantiert werden kann.

Die folgende Aufzählung erklärt einige Sicherheitsrisiken bei der browserbasierten Datenverarbeitung:

2.7.4.1. Client Konfiguration

Damit der Browser Daten sicher verarbeitet, muss er korrekt konfiguriert sein. Hierzu zählen die Konfiguration des Verschlüsselungsgrades (SSL), der Sicherheit von Scripting- und Websprachen, der Cookieeinstellungen und der Authentifizierungsverfahren. Falls mehrere Benutzer auf demselben Gerät arbeiten, ist auch eine benutzerdefinierte Konfiguration der Cache- und Historyeinrichtungen zu empfehlen. Außerdem sollten Passwörter nicht dauerhaft gespeichert werden.

Bei Terminaloptionen sollten ebenfalls benutzerdefinierte Verbindungen erstellt werden und die Verschlüsselung sollte aktiviert werden.

2.7.4.2. Sicherheitslücken im Browser

Im Vergleich zu lokal installierter Software befindet sich zwischen Betriebsystem und Anwendung eine weitere Schicht in Form des Browsers oder des Terminalclients. Aus Anwendersicht spielen diese Oberflächen eine wesentliche Rolle im ASP Betrieb. Gegen die Browser und dessen Technologien (Java, JavaScript, ActiveX, ICA, RDP, etc.) werden berechtigterweise immer wieder Sicherheitsbedenken erhoben. Hier eine kurze Zusammenfassung der wichtigsten Technologien:

- **HTML**

  Bis jetzt sind keine Sicherheitslücken bekannt.

- **Cookies**

  Keine unmittelbaren Sicherheitslücken, aber "Ausspähen" personalisierter Informationen durch DNS-Fälschung möglich.

- **JavaScript**

  Ursprünglich hatte JavaScript kein Sicherheitsmodell implementiert und sollte lediglich aufgrund eingeschränkter Funktionalität (keine Dateioperationen, keine Netzverbindungen) sicher sein. Jedoch gab es in der Geschichte der Internetbrowser immer wieder Sicherheitslücken in scheinbar harmlosen Funktionen, die Angriffe auf Daten ermöglichen. Dies ist ein Auszug der Sicherheitslücken von IE und Netscape:

Mögliche Aktion	**Betroffene Browser**
Fremde Dateien vom Clientgerät lesen (Datendiebstahl)	Netscape 4.5, 4.04 - 4.05 IE 4.0-4.01 und Vorversion 5.0
Ausspähen der zuletzt besuchten Seiten	Netscape 3.04, 4.07, 4.5
Ausspähen des Benutzernamens und Passwortes	Netscape, alle Versionen bis 4.04
Ausspähen des Inhaltsverzeichnisses der lokalen Festplatte	Netscape 2.0 und 2.01
Versenden von E-Mails ohne Kenntnis des Benutzers	Netscape 2.0 und 3.0

Tabelle 5 : Sicherheitslücken in Internetbrowsern

Es ist anzunehmen, dass viele der hier aufgeführten "Bugs" bereits beseitigt wurden, aber neue Fehler bereits wieder hinzugekommen sind.

- **Java**

  Da Java wesentlich reicher an Funktionalität ist, wurde es mit einem Sicherheits-Manager ausgestattet. Über eine in den Browsern eingebaute Sicherheitsvorschrift erlaubt dieser nur den Zugriff die legimitierten Ressourcen. Dies wird oft als das Java-"Sandbox"-Modell bezeichnet. Seit Version 4 von Netscape und Internet Explorer wurde die Technik der signierten Applets eingeführt. Diese bürgen für ihre Echtheit und ihren Ursprung, können aber eigene Sicherheitsvorschriften definieren. Stimmt der Benutzer der Ausführung eines

solcherart signierten Applets zu, erlaubt er demselben die beabsichtigten Funktionen durchzuführen. Das sich bei Signierung ergebende Sicherheitsniveau entspricht etwa dem von herkömmlicher, legal über den Handel erworbener Software.

Die Signierung ändert nichts an den bekannten Sicherheitslücken von Java:

Mögliche Aktion	**Betroffener Browser**
Absturz des Clientsystems	Netscape für NT Version 2, 3 und 4 IE für NT Version 3, 4 und 5
Ungültige Maschinebefehle ausführen	Netscape 2.0 und 2.01
Netzwerkverbindungen mit fremden Hosts herstellen	Netscape 2.0
Security Manager umgehen	Netscape 3.01 und früher IE 3.01 und früher

Tabelle 6 : Sicherheitslücken in Java VMs

Es gehen allerdings die Meinungen auseinander, ob Java oder JavaScript "gefährlicher" sei.

- **IE ActiveX und Netscape Plug-In**

  Bei ActiveX und Plug-Ins gibt es nur eine Signierungsfunktion. Es gibt keinerlei Sicherheitsrestriktionen. Stimmt man der Installation einer ActiveX- oder Plug-In-Komponente zu, so hat diese vollen Zugriff auf den Clientrechner. Aus diesem Grund muss ein besonders hohes Vertrauen zu dem Unternehmen bestehen, das die ActiveX oder Plug-In-Komponente anbietet.

- **Terminalclient**

  Die Terminalclients gibt es für den Browser als Java-Applet, ActiveX- sowie als Plug-In-Komponente oder als eigenständige Software. Sie verwenden derzeit entweder das ICA, RDP oder AIP-Protokoll. Laut [Paiha 2001] kann zumindest zu Citrix gesagt werden, dass bis jetzt noch keine Sicherheitslücken bekannt sind.

2.7.4.3. Erhöhte Funktionalität der Clientlogik

Die Anforderungen an die GUI bzw. an clientseitige Hard- und Softwareintegration sind bei ASP oftmals besonders groß. So verlangen z.B. die Terminalclients den vollen Zugriff auf alle Client-Ressourcen. Ebenfalls benötigen die Java Applets der websprachenbasierten Systeme oft Zugriffsrechte für das Dateisystem, den Drucker oder das Netzwerk.
Durch die höhere Komplexität steigt auch die Gefahr von Fehlfunktionen durch Sicherheitslücken.

2.7.4.4. Ständige Internetverbindung und unkontrollierbare Kommunikation

Bei ständiger Netzverbindung könnte die Wahrscheinlichkeit einer Übertragung von persönlicher Information "im Hintergrund" wahrscheinlicher sein. Das demonstrieren verschiedene Beispiele aus der Vergangenheit von internetbasierten Applikationen. Bei Realplayer, ICQ und Napster wurden Fälle bekannt, in denen ein "Ausspähen" persönlicher Daten geschehen ist, sei es durch die Applikation selbst oder durch Missbrauch eines anderen Benutzers ("Hacker"). Prinzipiell wären solche Szenarios auch bei ASP denkbar.

Der ASP ist jedoch nicht befugt, persönliche Daten (z.B. für Marketingzwecke) auszuspähen. Bei Citrix, WTS und Tarantella wird garantiert, dass über die ICA-, RDP- und AIP-Protokolle nur für den Terminalclient bestimmte Grafik- und Eingabeinformation ausgetauscht wird. Die Echtheit und der Ursprung der Daten bei websprachenbasierten Systemen und bei Terminalsystemen kann zudem über SSL garantiert werden.

Die ordnungsgemäße Übertragung ausschließlich für die Applikationsnutzung benötigter Information ist also dann gegeben, wenn die Clientprogrammlogik den vom Provider beabsichtigten Zweck sicher erfüllt.

Abschließend lässt sich über die Client-Sicherheit behaupten, dass es sich hier hauptsächlich um eine Vertrauenssache handelt. Der ASP kann mittels der SSL-Technologie für den Ursprung, die Echtheit, die Unverfälschtheit und Sicherheit seiner Inhalte garantieren. Vorsicht ist jedoch bei ungesicherten, kostenlosen Diensten gegeben. Ebenfalls sollten nur signierte Java-, ActiveX- und Plug-In-Komponenten installiert werden, für die der Urspung eindeutig hervorgeht. Selbiges trifft auf die Internetbrowser und die eigenständigen Terminalclients zu. Sie sollten ausschließlich von den offiziellen Webseiten heruntergeladen werden oder von den Original-CDs installiert werden.

Kapitel 3

ASP Basistechnologien in der Praxis

Wie bereits im theoretischen Teil dieser Arbeit festgestellt wurde, gibt es im wesentlichen zwei verschiedene Basistechnologien für ASP:

- Websprachenbasierte Systeme
- Terminalbasierte Systeme

In der ASP-Praxis haben beide Technologien ihre Vor- und Nachteile.

Bei websprachenbasierten Systemen gibt es seit kurzem, wie in den Kapiteln 2.2. und 2.6.1. erwähnt, nicht nur konventionelle 3-Schichten-Modelle auf Basis von *Web Servern*, sondern auch drei- oder vierschichtige Modelle mit *Application Servern* als sog. *Middleware*. Architekturen auf Basis von Application Servern sind im Hinblick auf ASP anders als Architekturen auf Basis "traditioneller" Internettechnologien zu bewerten.

Das erste Kapitel beschreibt die wichtigsten Internettechnologien und -sprachen aus der Praxis für ein konventionelles 3-Schichten-Modell auf Web Server-Basis.

Danach werden drei noch relativ neue Produkte auf Application Server-Basis beschrieben (Microsoft .NET auf Basis von COM+, IBM WebSphere und BEA WebLogic auf Basis von J2EE).

Zwei verschiedene terminalbasierte Systeme (Microsoft Windows Terminal Services, Citrix MetaFrame) werden ebenfalls auf "ASP-Tauglichkeit" untersucht.

3.1. Traditionelle Internettechnologien und -sprachen

"Traditionell" bedeutet in diesem Zusammenhang die Gesamtheit notwendiger Internettechnologien und -sprachen für den Einsatz im herkömmlichen 3-Schichtenmodell mit Client, Web Server (auch "Internet Server" genannt; mit entsprechenden Erweiterungen für serverseitige Internetsprachen) und Datenbankserver.

Nun sollen die derzeit wichtigsten, verbreitetsten und etabliertesten Internettechnologien- und -sprachen im Zusammenhang mit traditionellem webbasierten ASP vorgestellt werden.

Der ursprüngliche Zweck des WWW war die Navigation durch Informationsseiten. Diese Anforderungen wurden mit HTML, einer Beschreibungssprache für die Präsentation statischer Inhalte, umgesetzt. Mit dem Aufkommen dynamischer, personalisierter Inhalte und interaktiver Internetprogramme wie Webmail, Webshop und Chat mussten weitere Sprachen entwickelt werden, um den neuen Anforderungen gerecht zu werden. Außerdem mussten Möglichkeiten geschaffen werden, auf verteilte Ressourcen (Datenbanken, Internet) zuzugreifen. Das Application Service Providing stellt nun die höchsten Anforderungen an client- und serverseitige Internetsprachen und -technologien.

Eine Einteilung der zahlreich vorhandenen Internettechnologien kann nach vielen verschiedenen Kriterien erfolgen:

Prinzipiell gibt es vier Kategorien [Herzwurm 2001]:

- **Technologien zur Erstellung statischer Webinhalte (Beschreibungssprachen)**

  Hier gibt es **Auszeichnungssprachen** (HTML**,** XML) und **Formatierungssprachen**. (CSS, XSL). Auszeichnungssprachen dienen der Beschreibung eines Dokuments in syntaktischer oder inhaltlicher Hinsicht, nicht (wie Programmiersprachen) der Umsetzung eines Algorithmus. Es gibt keinen Kontroll- oder Datenfluss. Formatierungssprachen dienen dazu, bestimmten Bereichen eines Dokuments bestimmte Darstellungsformate (z.B. Abstände, Zeichenformate, Ausrichtungen) zuzuweisen. Formatierungsdokumente sind nicht eigenständig, da sie keine Daten enthalten. Sie geben nur an, wie bestimmte andere Dokumente anzuzeigen sind.

- **Technologien zur Erstellung dynamischer Webinhalte (Mechanismen, Programmiersprachen und Skripte)**

  An Internet-Programmiersprachen gibt es **interpretierte** (JavaScript, Perl über CGI) und **kompilierte** Sprachen (C oder C++ über CGI) sowie **Mischformen** (Kompilation bei erster Anfrage, Echtzeitkompilation bzw. Interpretation: Java Servlets).

  Sprachen, die sich in andere Sprachen (z.B. in HTML) **einbetten** lassen, werden auch **Skriptsprachen** oder kurz **Skripte** (JavaScript, ASP, JSP, PHP) genannt.

  Andere eigenständig oder extern ablaufende Programme werden schlicht **Programmiersprachen** (C, C++ und Perl über CGI; Servlets) genannt.

  Eine klare Trennung lässt sich außerdem zwischen (dynamischen) **clientseitigen** (JavaScript, JScript, Java Applets, Visual Basic / ActiveX, Plug-In) und **serverseitigen Sprachen und Mechanismen** (CGI / Perl, ASP, PHP, JSP, Servlets) vornehmen. Die Einteilung der dynamischen Internetsprachen in letztgenannte Kategorien erscheint im Hinblick auf die für ASP relevanten Anforderungen und Funktionen am geeignetsten.

- **Interprozesskommunikation (siehe auch Kapitel 2.2.)**

  Fast alle der verfügbaren serverseitigen Internetsprachen unterstützen Mechanismen zur Programmierung von verteilten Anwendungen. Sind solche vorhanden, werden sie jeweils bei der anschließenden Diskussion der Internetsprachen erwähnt.

  Die Aspekte von modernen Interprozess- und Komponentenarchitekturen wie COM, DCOM, COM+, CORBA und EJB werden gesondert in eigenen Kapiteln, beispielhaft anhand der notwendigen Application Server-Produkte erläutert, da Application Server vollkommen veränderte Qualitäten für webbasiertes ASP mit sich bringen.

- **Datenbankschnittstellen und -sprachen (siehe auch Kapitel 2.2)**

  Die wichtigsten Datenbankschnittstellen sind ODBC (für Windows) und JDBC (für Java). Die bedeutenste Datenbanksprache ist SQL.

  Datenbanktechnologien sind inzwischen sehr ausgereift, und die Technologien sind weder rein ASP-spezifisch, noch haben sich die Anforderungen an Datenbanksysteme durch das ASP geändert. Deshalb wird auf eine gesonderte Ausführung oben genannter Technologien verzichtet. Auch hier gilt jedoch, dass Application Server völlig neue Maßstäbe in Sachen Datenbankanbindung setzen.

Es folgt eine Diskussion der für ASP wichtigsten Internetsprachen aus der Praxis. Als Abschluss der Beispiele wird jedes Mal die jeweilige Bedeutung für ASP bewertet.

3.1.1. Beschreibungssprachen

3.1.1.1. Auszeichnungssprachen

- **HTML (Hypertext Markup Language)**

  HTML wird vom W3-Konsortium (W3C) normiert und ist von SGML (Standardized Generalized Markup Language) abgeleitet. HTML ist "die" Internetsprache für die Präsentation von Inhalten in Browsern. Der ursprüngliche Grundgedanke von HTML war, eine Navigation durch vernetzte Information zu ermöglichen [app2web 2002]. Dabei wurde auf die Einbindung vieler verschiedener Datenformate (Bilder, Texte etc.) und auf Plattformunabhängigkeit Wert gelegt. Für eine einfache, interaktive Inhaltsselektion wurden, neben den Links, folgende GUI-Elemente ("Formulare") integriert [Münz 2001]: Ein- und mehrzeilige Eingabefelder (auch für Passwörter), Auswahllisten (Ein- oder Mehrfachauswahl möglich), Radiobuttons, Checkboxen, Buttons und Datei-Upload-Buttons. Außerdem lassen sich Tastenkürzel und Tabulatorreihenfolgen für Formulare definieren. Alle anderen Sprachelemente von HTML dienen ausschließlich der Festlegung der Dokumentstruktur und des Layouts.

  Da HTML nicht für die Anbindung an Applikationen geplant war, ist das Angebot an GUI-Elementen sehr dürftig. Es gibt z.B. keine Pull-Down-Menüs und auch keine Möglichkeit, beliebige neue Fenster zu öffnen. Es gibt jedoch die Möglichkeit, clientseitige Skripts und Programmiersprachen in den HTML-Code einzubetten oder einzubinden.
  HTML ist heute die Grundlage für die formatierte Anzeige von Inhalten von webbasierten ASP-Applikationen. Das Angebot an Funktionalität und GUI-Elementen ist jedoch so spartanisch, dass es kaum ASP-Applikationen gibt, die ohne dynamische clientseitige Sprachen auskommen.

- **XML (eXtensible Markup Language)**

  XML ist ebenso wie HTML eine Teilmenge von SGML und vom W3C normiert [W3C 2000b]. XML dient der Kennzeichnung der semantischen Struktur eines Dokuments. XML selbst ist eine "Meta-Sprache", es dient lediglich zur Definition von neuen Sprachen, sogenannten XML-Vokabularen. Diese werden mithilfe der DTD (Dynamic Type Definition) definiert und legen jeweils die Menge der gültigen Elemente fest [Herzwurm 2001]. XML besitzt selbst keine Funktionalität, ist also "noch statischer" als HTML und beschreibt lediglich die Art von Daten. Dies prädestiniert XML als Format für jeglichen Datenaustausch zwischen heterogenen Systemen. Es gibt DTDs für mathematische Notationen, für Vektorgrafiken, für chemische Formeln und insbesondere für den elektronischen Austausch von Geschäftsinformationen [Herzwurm 2001].

  Im ASP-Bereich wird XML vornehmlich auf Serverseite verwendet, da der ASP dem Client Applikationen und nicht bloß Daten zur Verfügung stellt.

  Mittels der Formatierungssprache XSL können jedoch XML Dokumente in andere Formate wie HTML oder WML gewandelt werden, die vom Client verwendet werden können (siehe "XSL" unter "Formatierungssprachen").

  Zudem wird es für Interprozesskommunikation (Web Services, siehe Kapitel 4) und für die plattformunabhängige Gestaltung von Konfigurationsdaten für Webdienste verwendet.

  Auf Clientseite wird XML bislang nur vom Microsoft Internet Explorer ab der Version 5.0 unterstützt ... , welcher XML Daten als strukturierten Text anzeigt.

  Ein gänzlich neues Konzept sieht vor, XML als Beschreibungssprache für Benutzerinterface-Elemente zu verwenden, die vom Client mittels eines Java Applets

dargestellt werden [Herzwurm 2001]. Das Konzept wurde jedoch noch von keinem industriellen Hersteller umgesetzt.

3.1.1.2. Formatierungssprachen

Es gibt unterschiedliche Sprachen zur Formatierung von HTML und XML-Dokumenten [Herzwurm 2001][app2web 2002]:

- **CSS (Cascading Style Sheets)**

  CSS legt das Layout von HTML-Dokumenten (z.B. Farben, Abstände und Schriftstile) fest. CSS ermöglichen die Trennung der Benutzerinterfacelogik in Dokumentstruktur und Layout und eine zentrale Festlegung der Layoutdefinition für ganze Websites.
  Die Benutzerinterfacelogik von ASP-Applikationen muss durch die Seitenbasiertheit von Browsern meist auf sehr vielen Webseiten verteilt werden. Für ASP bedeutet dies, dass es mittels CSS eine einfache Möglichkeit gibt, der Benutzerschnittstelle ein über die gesamte Applikation einheitliches Aussehen zu verleihen.

  Seit 1998 existiert die zweite Version des CSS-Standards. Aktuelle Browser haben jedoch immer noch Probleme damit, zumindest Konformität mit CSS 1.0 zu erreichen [Herzwurm 2001].

- **XSL (eXtesible Stylesheet Language)**

  Auch XSL ist von W3C normiert. XSL dient der Umwandlung von XML-Dokumenten in andere Formate (z.B. HTML) mittels Filtern und Sortieren der enthaltenen XML-Daten.

  Für ASP ergibt sich daraus eine interessante Möglichkeit: Mittels XSL kann das Erscheinungsbild von XML Dokumenten, abhängig davon auf welchem Ausgabegerät diese dargestellt werden sollen, festgelegt werden. Besonders interessant erscheint diese Möglichkeit für die derzeit heterogenen Beschreibungssprachen für Desktops (HTML) und mobile Geräte (WML). Ein solches Verfahren wird leider noch von keinem Hersteller großzügig unterstützt, sodass entsprechende Transformationsvorschriften händisch implementiert werden müssen.

3.1.2. Clientseitige Internetsprachen

Erst clientseitige Internetsprachen (im folgenden zu **csIs** abgekürzt) erlauben die für ASP-Applikationen sehr wichtige Interaktivität (Dynamik) umzusetzen. Clientseitige Sprachen (Browsersprachen) erlauben eine dynamische Änderung von Anzeigeinformation für eine aktuelle Web-Seite ohne Serverinteraktion und ohne Neuaufbau der Internetseite. Mittels csIs werden z.B. oft Eingabeprüfungen vorgenommen und bei Nichtentsprechung Warnmeldungen in Message-Boxen ausgegeben, die mittels clientseitiger Internetsprachen programmierbar sind.

Der Lebenszyklus eines clientseitigen Internetprogramms ist immer auf die Internetseite beschränkt, von der das Programm abstammt. Beim Verlassen einer Seite ist der Programmprozess samt seinem Zustand verloren. Eine dynamische Anforderung von csIs-Funktionalität auf derselben Internetseite in Form von Modulen ist daher nicht möglich. Für ASP bedeutet dies, dass bei jeder Anforderung die Gesamtfunktionalität für den aktuellen Kontext geladen werden muss. Eine Ausnahme bilden hier nur die clientseitigen Mechanismen wie ActiveX und Plug-Ins, die Persistenz bieten.

CsIs müssen für websprachenbasiertes ASP verwendet werden, um dem hohen Interaktionslevel gerecht zu werden. Leider verschlechtert der Einsatz von csIs aber grundsätzlich die Wartbarkeit von ASP-Lösungen. Eine diesbezügliche Bewertung der

einzelnen csIs folgt im Anschluss an die jeweiligen Kapitel. Der Einfachheit halber sei darauf hingewiesen, dass als Quellen zu csIs durchwegs [app2web 2002] und [Herzwurm 2001] verwendet wurden.

3.1.2.1. Clientseitige Mechanismen

Durch clientseitige Mechanismen wird der Internet-Browser grafisches Front-End für beliebige Anwendungen, die zur Laufzeit oder einmalig aus dem Internet geladen werden. Clientseitige Mechanismen erlauben die Ausführung von Programmen, die in beliebigen Programmiersprachen geschrieben sein können, da sie als plattformabhängiger Binärcode heruntergeladen und ausgeführt werden. Sie stellen daher die performanteste Art von in den Browser integrierten Programmen dar, laufen jedoch mit vollen Rechten auf dem PC. Die Standardanwendungen dieser Verfahren sind Zusatzprogramme zur Anzeige von WWW-Inhalten, die mit den im Browser enthaltenen Funktionen nicht handhabbar sind. Verbreitete Beispiele sind Shockwave Flash, RealPlayer, RealAudio, Quicktime und PDF. Für ASP ist wesentlich, dass clientseitige Mechanismen die Schnittstelle zu terminalbasierten Systemen darstellen. Eine solche Schnittstelle (browserbasierter Terminalclient) muss, im Gegensatz dazu bei einem Einsatz für websprachenbasiertes ASP, nur einmalig heruntergeladen werden.

Für websprachenbasiertes ASP konnten sich clientseitige Mechanismen wegen der Plattformabhängigkeit, der langen Downloadzeiten und der fehlenden Sicherheitsmechanismen nicht durchsetzen.

- **ActiveX-Controls**

  ActiveX ist eine Microsoft Technologie und nur auf Windows-Plattformen (Intel) lauffähig. Grundsätzlich sind ActiveX-Controls auch nur im Internet Explorer lauffähig, es existiert zwar ein Plug-In für Netscape, das aber Inkompatibilitäten aufweist [app2web 2002]. Interessant macht sie die Tatsache, dass sie auch als Komponenten eingesetzt werden können, die bidirektional mit dem Rest der Webseite interagieren können. Der Aufruf der Komponentenfunktionalität wird dabei in der Regel durch Skriptsprachen bewerkstelligt.

  Die fehlende Beschränkung des Zugriffs auf das System wird bei ActiveX-Controls durch Zertifikate abgemildert, die den Urheber des Controls sicherstellen sollen.

  Für die wichtigste Terminaltechnologie, Citrix ICA, gibt es eine ActiveX-Control.

- **Plug-Ins**

  Plug-Ins sind eine Einrichtung von Netscape, ActiveX-Controls werden jedoch, nicht ganz korrekt, auch oft als "Plug-Ins" bezeichnet. Beim Aufruf eines entsprechenden Plug-Ins (per MIME-Type) übernimmt der Browser nur noch die Übergabe von Parametern. Dem Plug-In steht eine API des Web-Browsers zur Verfügung, um dessen Funktionalität zu nutzen (z.B. Anzeige, Streaming von per URL adressierten Ressourcen, etc.). Eine Interaktion mittels Skriptsprachen ist nicht möglich.

  Einige Programmiersprachen wie z.B: Shockwave, Flash oder spezielle Java-VMs sind als Plug-In verfügbar. Shockwave und Flash werden allerdings für ASP sehr selten eingesetzt. Als eine weitere wichtige generalisierte Maschine zur GUI-Visualisierung existiert ein Plug-In von Citrix für terminalbasiertes ASP.
  Für applikationsbezogene Funktionalitäten werden Plug-Ins nicht eingesetzt.

3.1.2.2. Clientseitige Skriptsprachen

Clientseitige Skriptsprachen dienen als Ergänzung zur HTML-Sprache zur Dynamisierung von WWW-Inhalten, Animation, und zu guter Letzt für ASP sehr wichtig, von Interaktion.

Skriptsprachen sind meist interpretierte, extra für den Einsatz in Browsern entwickelte Sprachen und werden durch spezielle Tags in HTML-Dokumente eingebettet, vom Server vollständig zum Browser übertragen und dort eigenständig vom Browser ausgeführt. Skripte werden ereignisgesteuert aufgerufen und können ihrerseits den Browser steuern, indem sie z.B. Seiten anzeigen lassen, die Werte von Formularen setzen oder Fenster öffnen.

Die Abbildung des Browsers erfolgt über das Objektmodell DOM (Document Object Model). Auf die Eigenschaften und Methoden der Objekte dieses Modells können die Skripte zugreifen.

Die drei wichtigsten Vertreter dieser Kategorie sind JavaScript, das von Netscape entwickelt wurde und in deren Browser integriert wurde, sowie Jscript und VBScript.

Im Vergleich zu HTML konnten sich keine oder nur wenig mächtige Standards etablieren. Daher müssen stets verschiedene Versionen für die verschiedenen Kombinationen von Browser und Plattform der Web-Anwendungen erstellt und gewartet werden.

- **JavaScript (ECMAScript)**

  JavaScript ist derzeit die weitverbreitetste Browser-Skriptsprache. JavaScript wurde ursprünglich von Netscape für deren Browser entwickelt. Auch Microsoft entwickelte für den Internet Explorer eine JavaScript ähnliche Sprache namens "Jscript". Bald wurde die Schnittmenge beider Sprachen zu dem sog. ECMA-Standard erhoben, der meist nicht ganz korrekt einfach mit "JavaScript" bezeichnet wird. Auch in dieser Diplomarbeit wird die Bezeichnung durchgängig so gehandhabt.

  Leider konzentrieren sich sowohl Netscape also auch Microsoft nicht auf die gemeinsame Weiterentwicklung des ECMA-Standards (derzeit in der Version 262), sondern erweitern jeweils ihre proprietären Technologien mit untereinander inkompatiblen Zusatzfunktionalitäten.

  JavaScript ist eine interpretierte, untypisierte und prototypenbasierte (d.h. es gibt keine Unterscheidung zwischen Objekten und Klassen) Sprache. Obwohl es interpretiert wird, ist es aufgrund seiner Einfachheit relativ schnell.

  Es reichert HTML um Funktionen wie Kontrollfluss (Bedingungen, Schleifen), Rechenfunktionen, Stringoperationen (reguläre Ausdrücke), Eingabevalidierung, Message-Boxen, Fensterverwaltung usw. an.

  Einige GUI-Elemente, die von den Betriebsystem-APIs bekannt sind, wie Pull-Down-Menus, Rich-Text-Boxen und Registerreiter fehlen jedoch.

  Zudem sind keine Funktionen vorhanden, die den Datenschutz des Benutzers oder die Datensicherheit des Endgerätes in irgendeiner Weise gefährden könnten. So sind keinerlei Dateioperationen und auch keine Netzwerkverbindungen möglich (siehe Kapitel 2.7.4., "Client Sicherheit").

  Websprachenbasiertes ASP ist ohne Einsatz von JavaScript kaum denkbar, und stellt bei gewissenhaftem Einhalten der ECMA-Spezifikation auch betreff Plattformunabhängigkeit und Sicherheit kein Problem dar.

- **VBScript**

  VBScript ist eine proprietäre Browser-Skriptsprache von Microsoft, die nur im MS Internet Explorer Verwendung findet und ungefähr dieselbe Funktionalität wie JScript zur

Verfügung stellt. VBScript erlaubt den Zugriff auf Microsofts Komponententechnologien, insbesondere auf die in die Webseite integrierten ActiveX-Komponenten.

VBScript hat den Ruf, dass mehr Sicherheitslücken als in JavaScript vorhanden sind.

Da VBScript nicht browserunabhängig ist, ist ein Einsatz im ASP-Umfeld ungeeignet.

3.1.2.3. Clientseitige Programmiersprachen

An clientseitigen Programmiersprachen, d.h. Sprachen, die nicht in HTML eingebettet sind, konnte sich bis jetzt nur eine Technologie einen Namen machen:

- **Java Applets**

  Java ist eine mächtige und vollständig objektorientierte Programmiersprache, mit vielen Ähnlichkeiten zu C++ und Smalltalk. In WWW-Browsern integrierte Interpreter (Java Virtual Maschine (JVM)) ermöglichen es, Java Applets zur Laufzeit aus dem Netz zu laden und auf dem lokalen Rechner auszuführen. Dies ist möglich, da Java Applets (so wie alle Java Klassen) in Form eines vorkompilierten, plattformunabhängigen, sog. Bytecodes ausgeliefert werden. Java Applets werden in einer sog. "Sandbox" ausgeführt, die unter Sicherheitsaspekten problematische Aktionen verhindert. Die Beschränkungen können allerdings auf vom Benutzer feingranular aufgehoben werden. Theoretisch sind Applets auf jeder Hardware- und Betriebsystemplattform einsetzbar, auf der eine JVM läuft. In alle gängigen Browser ist die JVM samt den grundlegenden APIs schon integriert.

  Die Sprache Java bietet reichhaltige Möglichkeiten, deren Nennung den Rahmen der Diplomarbeit sprengen würde. Für ASP ist wichtig, dass es zahlreiche Möglichkeiten zur GUI-Programmierung und eine gute Unterstützung für Internet-Protokolle bietet.

  Java Applets konnten sich trotzdem im ASP Sektor noch nicht durchsetzen. Im wesentlichen gibt es drei Gründe hierfür:

  1) Durch den Einsatz von Java-Applets erhöht sich der Design-, Implementier- und Wartungsaufwand.
  2) Niedrige Performance (Starten der JVM, lange Ladezeiten, höherer Ressourcenbedarf)
  3) Entwicklerplattformen für ASP-Software enthalten kaum Funktionen zur Integration von Applets.

  Nebenbei wäre noch anzumerken, dass auch Java (genau wie JavaScript), im Unterschied zu HTML, mit kleineren Kompatibilitätsproblemen und Sicherheitsproblemen zu kämpfen hat.

Fazit zu clientseitigen Internettechnologien:
An clientseitigen Techniken hat sich im ASP-Bereich, neben HTTP und HTML, die Skriptsprache JavaScript (genauer: "ECMAScript 262") etabliert. Sie ist schnell, relativ sicher und bei vernünftigem Einsatz auch zu vielen Browsern kompatibel.

3.1.3. Serverseitige Internetsprachen

Die Quellen zu den folgenden Informationen zu serverseitigen Internetsprachen stammen durchwegs von [Herzwurm 2001], [Hußmann 2001], [Sattler 2001], [Wurzenberger 2000], [Fischer 2001], [Saly 1999], [app2web 2002] und [Plessl 2001] und wurden der Übersichtlichkeit halber hier vorweggenommen.
Der Sinn der serverseitigen Sprachen ist, Individualisierung, Applikationsfunktionalität und Anbindung an Datenbanken zu erreichen. Aufgabe der serverseitigen Techniken ist es, die

verschiedenen Datenquellen (Datenbanken, Dateisysteme, ERP-Systeme) und Teile der Anwendungslogik der Web-Applikationen zu integrieren und passende HTML-Ansichten zu generieren, sowie die Benutzereingaben in entsprechende Aufrufe der Anwendungslogik beziehungsweise Modifikationen der Datenquellen umzusetzen.

Da die HTML- bzw. HTTP-Standards keine Möglichkeiten zur serverseitigen dynamischen Inhaltsgenerierung oder -formatierung vorsehen, sind im Laufe der Zeit beginnend mit dem Common Gateway Interface (CGI) entsprechende Techniken als Ergänzung entstanden.

3.1.3.1. Serverseitige Mechanismen

Bei den serverseitigen Mechanismen für Websprachen werden ähnlich wie bei den clientseitigen Mechanismen, externe Maschinenprogramme ausgeführt.

- **CGI (Common Gateway Interface)**

  CGI ist die älteste und bewährteste Technik für die Generierung dynamischer Webseiten.

  CGI ist ein Standard zur Definition einer Schnittstelle zwischen Webserver und externen Programmen. Die Programme können dabei in jeder beliebigen Programmier- oder Skriptsprache implementiert sein. Die Programme können hierbei in ausführbaren Binary-Code vorliegen oder, bei Interpretersprachen, im Quelltext, dann ist allerdings zusätzlich ein externer Interpreter erforderlich. Die bekanntesten Sprachen sind Unix Shell, Phyton, Tcl, Perl, C, C++, Java, Fortran und Visual Basic. Die populärste und verbreitetste CGI-Sprache ist Perl.

  CGI-Programme befinden sich standardmäßig in einem speziellen Verzeichnis (z.B.: "cgi-bin") und werden anhand ihrer Dateiendung als solche erkannt (z.B: ".cgi" oder ".pl").

  Eine Übergabe von Parametern an die Programme erfolgt über HTTP-POST oder HTTP-GET durch den Webserver mittels Umgebungsvariablen, den Standardeingabekanal (stdin) oder auch über Kommandozeilenparameter. Die Ausgabe der Programme an den Standardausgabekanal wird vom Webserver an den Client weitergeleitet. Es können prinzipiell beliebige Dateitypen erzeugt werden (z.B. HTML, XML, Texte).

  Perl stellt Möglichkeiten zur Datenbankanbindung bereit (ODBC), jedoch fehlen Mechanismen zur Kommunikation mit verteilten Objekten.

  Bedingt durch seine Flexibilität kann ein unsauber konfiguriertes CGI ein ernstes Sicherheitsrisiko sein. Außerdem wird für jedes externe Programm ein eigener Prozess gestartet und damit werden unverhältnismäßig viele Ressourcen gebunden.

  Zusätzlich bereitet die Skalierbarkeit Probleme, so ist es nicht möglich, die Rechnerlast über mehrere Rechner zu verteilen (Load-Balancing).

  Zudem werden keine Mechanismen zur Unterstützung von Sessions zur Verfügung gestellt.

  Auch im ASP Bereich wird oftmals noch auf CGI-Lösungen zurückgegriffen, das Modell stellt jedoch aus eben genannten Gründen ein Auslaufmodell dar [Paiha 2001]. Die für hohe Service Levels wichtigen Anforderungen an Sessionverwaltung, Skalierbarkeit, Sicherheit und Performance werden durch CGI nicht gedeckt.

3.1.3.2. Serverseitige Skriptsprachen

Serverseitige Skriptsprachen sind zwar ebenso wie clientseitige Skriptsprachen in den HTML-Code eingebettet, funktionieren jedoch etwas anders: Vor der Zustellung der HTML-Seite an den Browser werden die jeweiligen Skriptabschnitte durch entsprechende Interpreter ausgewertet und die Ergebnisse, üblicherweise HTML- oder XML-Tags, durch die

Skriptabschnitte ersetzt. Dadurch erhält der Client nur HTML (optional mit clientseitigen Skripten, Programmiersprachen oder Mechanismen angereichert).

- **SSI (Server Side Includes)**

  Unter SSIs versteht man in HTML-Kommentaren untergebrachte Anweisungen, die vor der Auslieferung einer HTML-Datei an den Browser durch den Webserver interpretiert und ausgeführt werden, um zusätzlichen Inhalt in die auszuliefernde HTML-Datei einzubinden. Beispiele für solche Anweisungen sind:

  - Ausgabe von Umgebungsvariablen wie Namen der Datei
  - Datum der letzten Modifikation
  - Aufnahme anderer HTML-Dateien: durch diesen Mechanismus ist die Wiederverwendung von Inhalt mit wenig Aufwand möglich, wie es bei einer Gestaltung einer größeren Webapplikation mit einheitlichem Layout erforderlich ist
  - Ausführung von Programmen über CGI

Viele Webserver unterstützen SSI, die zuerst im NCSA-Webserver vorhanden waren. Ein serverübergreifender Standard existiert indes nicht.

ASP Applikationen, die HTML-Module oder CGI verwenden, werden durch SSI unterstützt.

- **JSP (Java Server Pages)**

  Java Server Pages sind ein auf Servlets (siehe "serverseitige Programmiersprachen") aufbauender Mechanismus, der die Einbettung von Java über XML-ähnliche Tags in HTML erlaubt. Dazu kombiniert ein JSP Dokument Standard-HTML mit speziellen JSP Elementen, die von der sog. JSP Engine vor dem Senden des Dokuments verarbeitet und durch entsprechendes HTML ersetzt werden. JSP haben die Dateiendung ".jsp".

  In JSP können, ähnlich wie bei SSI, Komponenten eingebunden werden, die mit "Java Beans" bezeichnet werden. Dies sind wiederverwendbare Softwarebausteine mit Eigenschaften, die gesetzt bzw. ausgelesen und in HTML-Seiten eingebunden werden können. JSP im Zusammenhang mit Java Beans erlauben bei konsequenter Programmierung eine klare Trennung von Benutzerinterfacelogik (mittels JSP implementiert) und Applikationslogik (mittels Java Beans implementiert). Außerdem wird dadurch die für ASP sehr wichtige Skalierbarkeit sichergestellt und Load-Balancing ermöglicht.

  Java Server Pages bieten den vollständigen Java-Sprachumfang und werden vor dem erstmaligen Aufruf bzw. bei der Veröffentlichung in Servlets übersetzt. Von diesem technischen Standpunkt aus betrachtet sind JSP nichts anderes als ein Servlet-Generator, der die Erzeugung von Servlets mit umfangreicherer HTML-Ausgabe erleichtert.

- **ASP (Active Server Pages)**

  ASP ist die direkt mit JSP konkurrierende Technologie von Microsoft. ASP ist, im Gegensatz zu ASP.NET, nur quasi-plattformunabhängig. ASP-Dokumente haben die Dateiendung ".asp". Auch hier gibt es spezielle Tags, Scripting und den Zugriff auf Serverkomponenten (über COM). Der Vorteil von ASP ist die Sprachenunabhängigkeit. Es werden serverseitiges JScript, VBScript und Perl unterstützt. Der Nachteil von ASP (im Unterschied zu ASP.NET) ist die schlechtere Performance, da Quellcode interpretiert werden muss.

Sessionverwaltung wird von ASP sehr gut unterstützt, indem es für die Anwendung (für alle Sitzungen), die jeweilige Sitzung und die aktuelle Seite entsprechende Informationen bereithält (sog. serverseitige Sessionvariablen).

Eine Anbindung an verteilte Komponenten ist mittels COM oder DCOM möglich.

Bei Bedarf lassen sich zusätzliche Objekte installieren, die z.B. die Eigenschaften des Browsers ermitteln oder Funktionen zur Anbindung an Datenbanken oder sogar an Dateisystem bieten. Dies sind interessante Ansätze für Application Service Providing.

- **PHP (Hypertext Preprocessor)**

PHP, ursprünglich "Personal Homepage Tools", ist eine kostenlose Open-Source-Skriptsprache für das Web, die ähnlich ASP oder JSP über Tags in HTML eingebettet und auf dem Web Server ausgeführt wird. PHP-Dokumente haben die Endung ".php" oder ".php3". Die Syntax verbindet Elemente von C, Java sowie Perl und bietet Unterstützung für objektorientierte Programmierung inkl. Vererbung. Durch die Sprache werden außerdem alle gängigen Datenbanksysteme unterstützt sowie Funktionen zur Sitzungsverwaltung zur Verfügung gestellt (die den von Active Server Pages sehr ähnlich sind). PHP erlaubt den direkten Zugriff auf COM-Komponenten, Java-Klassen und Enterprise Java Beans. PHP-Skripte werden ähnlich wie JSP vor der Ausführung komplett analysiert und übersetzt. Der Einsatz vom PHP ist nicht auf bestimmte Web Server gebunden, am häufigsten ist momentan aber die Kombination mit Apache und mySQL auf Linux anzutreffen, man spricht dann abgekürzt von einem *LAMP*-System (Linux-Apache-mySQL-PHP). Im Apache-Web Server ist der PHP-Interpreter integriert.

3.1.3.3. Serverseitige Programmiersprachen

Eine Möglichkeit, beliebige Programmiersprachen zur HTML-Generierung zu verwenden, wurde ja bereits genannt (CGI). Es gibt allerdings eine weitere, an die Sprache "Java" gebundene Variante, die die Nachteile von CGI aufhebt:

- **Servlets**

Servlets sind Java-Objekte, die im Kontext eines Web-Servers ablaufen. Dazu gibt es für die gängigen Web Server (z.B. Apache) spezielle Java-Interpreter-Module (z.B. "JRun" für IIS oder "JServ" oder "Tomcat" für Apache), oder die Servlet Engine ist direkt im Web-Server integriert. Einige Vorteile gegenüber CGI-Skripts sind die Plattformunabhängigkeit, der Zugang zu allen Java APIs, die größere Performance, die eingebauten Funktionen zur Sitzungsverwaltung, zur Interprozesskommunikation (RMI) und zur Datenbankanbindung (JDBC) und die Möglichkeiten zum Load-Balancing aufgrund der Modularisierungsmöglichkeiten.

Folgende Feststellungen gelten aber auch für CGI: Servlet-Programme haben im Unterschied zu JSP einen strukturierteren Aufbau und bieten bessere Möglichkeiten zur Modularisierung. Sie sind geeigneter, wenn die Applikationsprogrammlogik der Benutzerinterfaceprogrammlogik vorrangig ist. Der Nachteil ist, dass keine HTML-Editoren (sog. "Autorensysteme") verwendet werden können. Bei umfangreichen Benutzerschnittstellen wird die Ausgabe der HTML-Tags über "Print"-Anweisungen rasch unübersichtlich. Servlets werden daher oft im Zusammenhang mit JSP verwendet.

Abschließend ein Überblick über die eben genannten Internetsprachen:

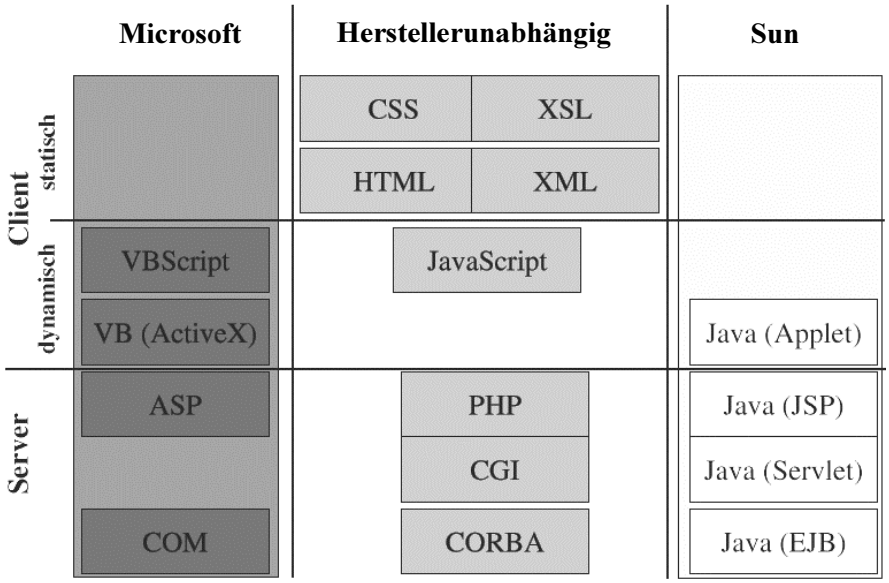

Abbildung 3-1 : Traditionelle Internetsprachen ([Herzwurm 2001], geändert)

In der linken Spalte befinden sich die Technologien vom Microsoft, in der Mitte die standardisierten bzw. Open-Source Technologien. Rechts befinden sich die Java-Technologien von Sun. Jede einzelne Zeile der Tabelle stellt in ihrer Funktion vergleichbare Internettechnologien gegenüber.

Folgende Tabelle vergleicht die traditionellen serverseitigen Internettechnologien hinsichtlich Leistungsfähigkeit und Qualität (zusammengestellt aus den Bewertungen von [Saly 1999], [Plessl 2001], [Herzwurm 2001] und [Fischer 2001]):

	CGI mit Perl	**JSP, Servlets**	**ASP**	**PHP3**
Skalierbarkeit	-	+	~	~
Sicherheit	-	+	~	~
Erlernbarkeit	+	-	+	+
Plattformunabhängigkeit	+	+	-	~
Trennung von Benutzerschnittstelle und Applikations-programmlogik	-	+	-	-
Entwicklungsumgebung	Nicht beurteilt	~	-	-
Sessionverwaltung	-	+	~	~ (PHP4: +)
Datenbankzugriff	Modul	JDBC	ODBC	Integriert
Verbreitung	+	~	+	+

+ ... gut
~ ... durchschnittlich
- ... schlecht

Tabelle 7 : Serverseitige Programmiersprachen im Vergleich

CGI zeigt, wie erwähnt, deutliche Schwachstellen bei Skalierbarkeit, Sicherheit und Sessionverwaltung. JSP und Servlets sind zwar sehr leistungsfähig, aber auch schwer zu erlernen. PHP und ASP bieten sehr ähnliche Funktionalität. PHP zeigt jedoch deutlichere Anzeichen von Plattformunabhängigkeit und ist zudem kostenlos bzw. Open-Source.

Die Frage nach der generellen Eignung von Internetsprachen für ASP muss jedoch gesondert behandelt werden.

3.1.4. Eignung für ASP

Websprachen-Interpreter und -Compiler formten, zusammen mit Web Servern und Datenbanken, die erste technische Basis für ASP.

Wie ersichtlich, ist die Auswahl an Internetsprachen mittlererweile sehr reichhaltig, sodass für jede einzelne Problemstellung die jeweils geeignetsten Sprachen ausgewählt werden können.

Die Flexibilität der Websprachen erlaubt es, verschiedene Technologien zu kombinieren. Grundsätzlich sind dadurch jedoch der **Wartungsaufwand** und die Wartungskosten von ASP-Lösungen auf Basis von traditionellen Websprachen höher als bei herkömmlichen Einzelplatz-Applikationen.

Besonders schwierig ist es, **Applikationsprogrammlogik** und **Benutzerinterface-programmlogik** gut zu trennen. Es erfordert ein klares Design und vielmehr noch eine Konsequenz, die aufgrund technischer Einschränkungen oft nicht durchführbar ist. Die Umsetzung der Benutzerinterfaceprogrammlogik geschieht in der Regel mittels HTML und JavaScript, da diese beiden Technologien hohe Performance und Kompatibilität zu vielen Browsern aufweisen. Werden nur diese beiden Technologien (neben Java Applets) eingesetzt, so ist damit die höchste **Geräte- und Plattformunabhängigkeit** zu erzielen.

Serverseitige Programme sind zumeist in Perl (über CGI) oder über Servlets realisiert. CGI ist jedoch ein Auslaufmodell [Paiha 2001]. Es ist für ASP aufgrund der schwachen Performance, Skalierbarkeit und Sicherheit eher ungeeignet. Der alleinige Einsatz von Servlets ist aufgrund der Unübersichtlichkeit und schlechten Wartbarkeit nur in Ausnahmefällen der Kombination JSP und JavaBeans oder JSP und Servlets vorzuziehen.

Verbreiteter und einfacher zu benutzen (unterstützt durch Autorensysteme) sind serverseitige Skriptsprachen. Die serverseitigen Skriptsprachen sind alle in ihrer Leistungsfähigkeit in etwa gleich. PHP, ASP und JSP, sie alle bieten **Sessionverwaltung**, **Datenbankanbindung, Interoperabilität** und **Mehrbenutzerfähigkeit** und somit **Personalisierungsmöglichkeiten**. ASP ist zusätzlich sprachenunabhängig, JSP ist hingegen plattformunabhängig. PHP ist plattformunabhängig und, als Open-Source-Sprache, kostenlos. Außerdem ist PHP performanter als ASP, da in PHP nur eine Sprache integriert ist. Java erscheint als typengebundene Sprache für das Design von dynamischen Benutzerschnittstellen möglicherweise weniger geeignet als die typenlosen Sprachen JScript, VBScript oder PHP.

Die **Kommunikation** zum Endgerät wird normalerweise über HTTP abgewickelt. Java-Applets können jedoch eigene Protokolle (z.B. RMI für entfernte Methodenaufrufe) benutzen.

Die **Performance** der Internettechnologien ist so verschieden wie die Funktionalität dieser Sprachen. Performance sollte nur bei vergleichbaren Technologien ein Auswahlkriterium sein. Von den serverseitigen Skript- und Programmiersprachen ist PHP die performanteste, von den clientseitigen ist es die Kombination aus HTML und JavaScript. HTML gemeinsam mit JavaScript ist auf jedem Fall schneller als temporäre ActiveX-Controls oder Java-Applets. Installierbare ActiveX-Komponenten und Applets können jedoch eingesetzt werden, wenn zumindest nur eine einmalige Installation pro Applikation notwendig ist.

Die **Benutzerfreundlichkeit** und **Interaktivität** ist in hohem Maße von der verwendeten Applikation und Technologien abhängig. HTML bietet nur sehr wenige Interaktionsmöglichkeiten, während JavaScript mehr Interaktvität bietet. Java Applets und ActiveX-Komponenten schließlich erreichen das höchste Interaktionsniveau.

Viele Mechanismen, die für hohe Service-Levels notwendig sind, wie **Skalierbarkeit**, **Hochverfügbarkeit**, **Zuverlässigkeit**, **Integrierbarkeit**, **Wartbarkeit**, **Sicherheit** (Authentifizierung) müssen sorgfältig entworfen werden und manuell implementiert werden.

Auch eine eventuelle Anpassung der Benutzerschnittstelle einer Web-Applikation an das Endgerät muss händisch implementiert werden. In vielen Fällen helfen kommerzielle oder frei verfügbare Softwarekomponenten. Nur sehr wenige Features werden von Web Servern (wie z.B. SSL-Verschlüsselung) zur Verfügung gestellt.

Fazit:

Websprachenbasierte Applikationen bieten Benutzerschnittstellen, die auf clientseitigen Browsersprachen und -mechanismen beruhen. Diese reichen nicht an die Leistungsfähigkeit von Benutzerschnittstellen von herkömmlicher Software heran. Meist bieten diese Applikationen für den Benutzer keinen Zugriff auf das Serverdateisystem, da der Aufwand für die Verwaltung und Bereitstellung zu groß wäre.

Für ASP auf Basis von niedrigen Service Levels (z.B. kostenlose Angebote) können weiterhin Lösungen mit traditionellen Webtechnologien eingesetzt werden. Denn der Aufwand für hohe Service Levels wie Skalierbarkeit, Hochverfügbarkeit, Zuverlässigkeit, Integrierbarkeit, Wartbarkeit und Sicherheit ist sehr hoch.

Microsoft .NET und die sog. *Application Server*, die in den nächsten Kapiteln vorgestellt werden, erleichtern die Umsetzung dieser Aufgaben, sind jedoch tlw. auch sehr teuer.

3.2. Microsoft .NET

Microsoft hat im Juni 2000 ein neues Konzept für eine komponenten- und internetorientierte Entwicklung ihrer Soft- und Hardwareprodukte bekannt gegeben. Es heißt "Microsoft .NET" und bezeichnet nicht ein einzelnes Produkt, sondern eine Strategie, die durch zwei Ausrichtungen gekennzeichnet ist [Schaub 2000]:

- Das Microsoft .NET Framework
- Die .NET Produkte und Services von Microsoft und Drittanbietern

Das .NET Framework wurde im letzten Quartal 2001 veröffentlicht. In Windows XP ist die .NET Funktionalität bereits integriert, für Windows NT4.0 bzw. Windows 2000 gibt es kostenlose Updates.

Microsoft .NET soll im folgenden nicht im vollen Umfang beschrieben werden, sondern nur die für Application Service Providing relevanten Aspekte zuzüglich anderen Informationen, soweit sie zum Verständnis des .NET Konzeptes notwendig sind. Detaillierte Informationen über das .NET Framework lassen sich in [Thai 2002] nachlesen.

3.2.1. .NET Konzept

Die Applikationslandschaft ist heute nach wie vor vielfach von Anwendungen geprägt, die nicht komponentenbasiert entwickelt wurden. Änderungen und Anpassungen können bei solchen Anwendungen auf Grund ihres starren Aufbaus und der fehlenden Flexibilität nur schwer vorgenommen werden.

Das .NET-Konzept hingegen ist auf die Entwicklung vollständig komponentenorientierter Applikationen ausgerichtet, wobei die zur Programmausführung benötigten Komponenten nicht ortsgebunden sind. Durch diese dezentrale Verfügbarkeit der über das Inter- und Intranet verteilten Module entsteht eine neue Art der Anwendungsentwicklung. Die Programmfunktionalität wird nicht mehr nur durch eine große Applikation zur Verfügung gestellt, sondern durch mehrere Module, die auf bestimmte Funktionalitäten spezialisiert sind,

ersetzt. Im Browser, der als Oberfläche und zentrale Ablaufumgebung dient, werden diese Module zu personalisierten Anwendungen zusammengefügt. Der Austausch von Daten erfolgt über die standardisierten Internetprotokolle und Markupsprachen, die Microsoft als alleinige Basis für das gesamte .NET-Konzept vorsieht. Somit ist .NET eindeutig den webbasierten Basistechnologien zuzuordnen.

Das gemeinsame standardisierte Datenformat **XML** (eXtensible Markup Language) [W3C 2000b] spielt hierbei innerhalb des .NET-Konzepts eine herausragend wichtige Rolle. Durch in SOAP (Simple Object Access Protocol) [W3C 2000a] realisierte Verknüpfungen können die unterschiedlichsten Anwendungen und Dienste XML-Daten miteinander austauschen.
Microsoft tritt durch das Anbieten solcher Dienste als Produzent und Hoster von webbasierenden Basismodulen auf. Auch als Publisher von Web-Diensten und -Anwendungen wie Office.NET will das Softwarehaus tätig werden. Somit beschränkt sich Microsoft nicht mehr nur auf das Anbieten von Entwicklungswerkzeugen, Middleware-Technologie und Betriebssystemen, sondern nutzt dieses neu geschaffene Geschäftsfeld als ASP und Anbieter von ASP Basistechnologien.

.NET ist viel modularer und enger mit diversen Komponenten aus dem Internet, den sogenannten **Web Services** (siehe auch Kapitel 4) verknüpft. Windows ist nicht mehr ein monolithisches System, sondern vielmehr kann sich jeder Anwender sein Windows nach Bedarf zusammenstellen.

Da .NET auf offenen Standards basiert, können auch andere Betriebsysteme auf diese Services zugreifen oder solche anbieten. Eine Portierung der öffentlichen Spezifikationen von .NET auf Linux ist bereits weit fortgeschritten [Mono 2002].

Eine kürzere Entwicklungszeit bei webbasierten Anwendungen erhofft sich Microsoft durch die konsequente Einhaltung der .NET-Prinzipien, wie z.B. ähnliche Klassendiagramme bei Win Forms (Windows-Systemklasse) und **Web Forms** (Webobjekte-Klasse), sprachübergreifende Vererbung, Garbagecollection, Sicherheit und Abwärtskompatibilität zu COM-(Component Object Model) Komponenten.

Die Kombination aus Web Forms, für die einfache Portierung der Benutzerschnittstellen ins Web, und Web Services, die eine modulare, internetbasierte, objektorientierte Programmierung erlauben, verkürzt die Umstellung herkömmlicher Software auf den ASP-Betrieb.

3.2.2. .NET Framework

Das .NET Framework ist die Laufzeitumgebung für die .NET Anwendungen. (Anmerkung zur Abbildung 3-2: In der Release-Version von .NET wurde die Bezeichnung "ASP+" in "ASP.NET" sowie "ADO+" in "ADO.NET" umbenannt.)

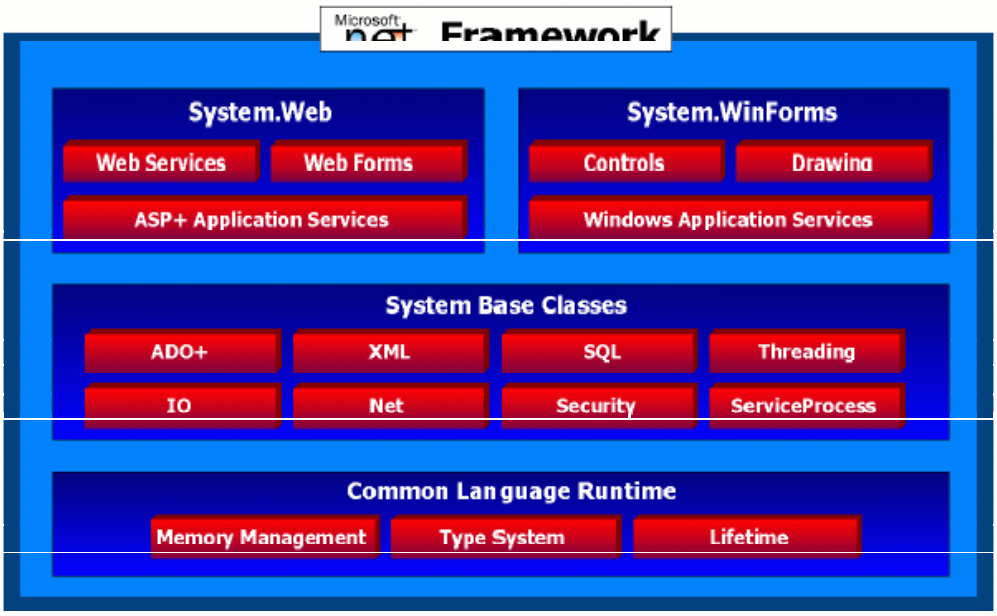

Abbildung 3-2 : Microsoft .NET Framework [Schaub 2000]

Für die Entwicklung von webbasierten Applikationen ist vor allem der System.Web-Namespace relevant. (In einem sog. "Namespace" können Klassen mit ähnlichen Einsatzgebieten zusammengefasst werden).

Der System.Win Forms Namespace hingegen enthält nur Klassen für die Programmierung (hauptsächlich der GUI) von Win32-Applikationen. Diese Komponente könnte für die Erstellung von spezialisierten ASP-Browsern (siehe Kapitel 2.1.3., "Vergleich von Terminals und Websprachen") oder für ASP, das auf einem internetbasierten Client-/Servermodell (d.h. Teile der Anwendungslogik sind auch am Client installiert) aufbaut, relevant sein. Microsoft will mit .NET auch diesen Markt der "Fat-Clients" erschließen (bisher im Geschäftsbereich dominiert von SAP u.a., im Consumer-Bereich von Napster, ICQ, etc.). Der Microsoft MSN Explorer [MSN 2002] ist ein Beispiel hierfür. In Zukunft soll sich auch MS Office der Web Services bedienen. Ob dies auch das Modell der Zukunft für ASP sein wird ist fraglich, da wieder, wie bisher, für jede Applikation Clientinstallationen notwendig werden. Aus diesem Grund, und weil die Win32-Applikationserstellung nicht ASP-spezifisch ist, werden die Win Forms hier nicht näher betrachtet.

Bevor Web Forms und Web Services beschrieben werden, einige Worte über die unterste Schicht des .NET Frameworks (Abbildung 3-2):

Dem gesamten Framework liegt die Common Language Runtime (CLR) zugrunde, die eine "virtuelle Maschine" (VM), ähnlich der JAVA-VM darstellt. Der Bytecode ("Managed Code") ist hierfür in der CPU-unabhängigen Maschinensprache "Intermediate Language" (IL) formuliert. IL kann mit Objekten, Methoden und Exceptions umgehen und hat einen Garbage Collector und Typensicherheit integriert.

Bei Webapplikationen liegt unter .NET nun die Applikationsprogrammlogik (z.B. in "C#" implementiert, eine neuartige, Java sehr ähnliche Sprache für .NET) in kompilierter Form vor, während die Benutzerschnittstellenprogrammlogik (mittels WebForms in ASP.NET) in Echtzeit mittels des "JIT" (Just-In-Time)- Compilers übersetzt wird. Dieses Verhalten spielt sowohl bei Web Applications (die Web Forms benutzen) als auch Web Services eine wichtige Rolle.

3.2.2.1. Web Forms

Die Web Forms werden auch als Active Server Pages (ASP+, ASP.NET; nicht zu verwechseln mit Application Service Providing !) bezeichnet und erlauben die Erstellung von Webseiten als Bestandteil einer Web Applikation.

Eine Art der "Komponentisierung" von Webseiten wird unterstützt: Sogenannte "Pagelets" erlauben ASP.NET Seiten in andere ASP.NET Seiten zu importieren. Damit können oft benötigte Teile der Benutzeroberflächenfunktionalität mehreren Websites zur Verfügung gestellt werden.

Die Entwicklung, Wartung und Integrierbarkeit wird durch ein ereignisbasiertes, serverseitiges Programmiermodell vereinfacht: Das "Code Behind" Feature von ASP.NET erlaubt, Design (Benutzerschnittstellenprogrammlogik) und Applikationsprogrammlogik zu trennen. Die Applikationsprogrammlogik (auch als "business logic" bezeichnet) wird von der IDL kompiliert. Wie bereits erwähnt, kommen hierfür bereits zahlreiche Programmiersprachen (C#, VB.NET, VC++) in Frage, da die IL sprachenunabhängig ist. Eine Aufteilung in Clientcode (Browser) und Servercode ist nicht mehr erforderlich, da .NET dem Programmierer eine rein serverseitige Programmierung erlaubt, bei der clientseitige Abläufe verborgen bleiben können. .NET nimmt dem Programmierer dabei eine Reihe von Problemen ab: Automatische Verwaltung von Sessioninformation (wahlweise serverseitig oder clientseitig via Cookies), Personalisierung, Caching und wahlweise verschlüsselte oder unverschlüsselte Übertragung. Jeglicher Client-Code wird automatisch zur Laufzeit generiert. Leider nur teilweise wird auch die Seitenorientierung des Browsers vor dem Programmierer verborgen.
So sieht der System.WebForms- Namespace aus:

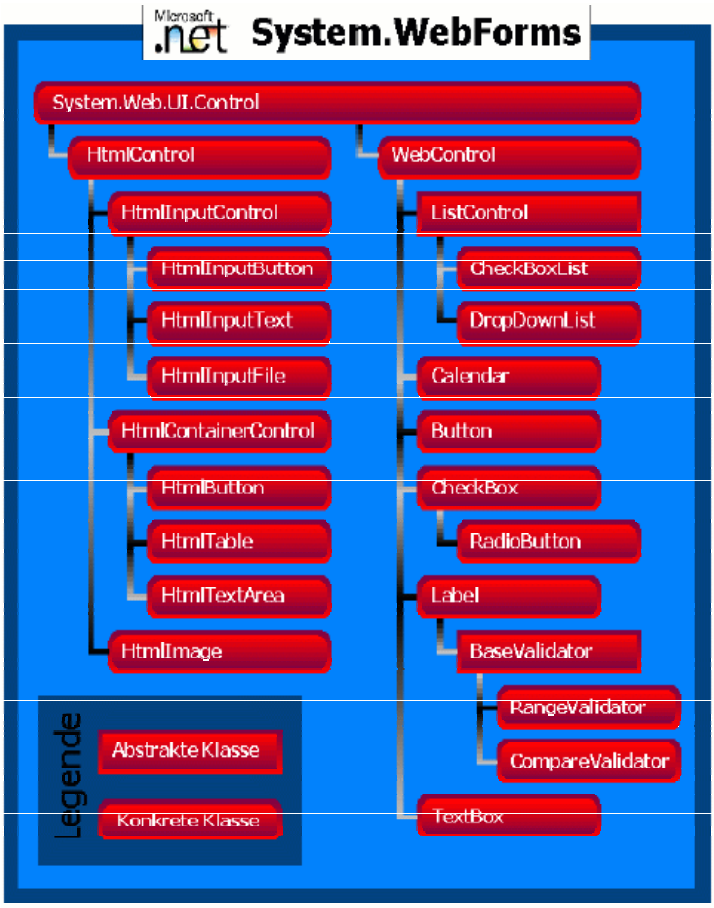

Abbildung 3-3 : Auszug aus dem Web Forms Klassendiagramm [Schaub 2000]

Gewöhnliche Rechtecke stellen abstrakte Klassen dar, während Rechtecke mit abgerundeten Ecken konkrete Klassen symbolisieren.

Für ASP ist vor allem die Untermenge der Web Controls interessant. Sie nehmen Rücksicht auf die Fähigkeiten und die Version des verwendeten Browsers, auf die HTML controls wird nicht näher eingegangen, da sie dazu nicht in der Lage sind.

Die Web controls beinhalten sowohl "traditionelle" Controls (z.B. TextBox und Button) wie auch Controls, die auf einem höheren Abstraktionslevel (z.B. Calender und DataGrid (Tabelle, die ihre Daten automatisch aus einer Datenbank bezieht)) angesiedelt sind.

Beim Einsatz von Web controls wird wahlweise HTML mit JavaScript oder VBScript oder sogar WML generiert. So enthalten z.B. die Controls zur Werteprüfung die Option Scripts zu generieren, die auf dem Client ausgeführt werden, um Interaktivität zu ermöglichen. Das System ist dabei so offen gestaltet, dass Erweiterungen jederzeit möglich sind. Jede Eigenschaft eines Controls kann auch zur Laufzeit aus einer Datenbank geladen werden.

Die Entwicklung von Web Controls hat erst begonnen, und es ist zu hoffen, dass sich auch viele Fremdhersteller daran beteiligen werden.

3.2.2.2. Web Services

Etwas anders als Web Applikationen verhalten sich Web Services. Bei Web Services handelt es sich um die Bereitstellung von Objektklassen über Netze, ein RPC (Remote Procedure Calling)-Konzept für das Internet. Web Services werden in .NET durch den Microsoft Webserver IIS (Internet Information Server) unterstützt.

Das Leistungsspektrum von Web Services umfasst:

- Geräte-, plattform- und sprachenunabhängige Bereitstellung von Funktionalität (Methoden) über das Internet
- Einsatz von standardisierten Protokollen: wahlweise HTTP oder SOAP (Simple Object Access Protocol; basiert auf XML mit HTTP als Transportprotokoll)
- Parameter via HTTP-GET, HTTP-POST oder SOAP, Rückgabewerte via XML oder SOAP
- Automatisch generierte HTML-Seite (http://*host*/*Webservicename*/*Webservicename*.asmx) zum Testen eines Web Services
- Selbstbeschreibend durch dynamisch generierte WSDL (Web Service Description Language; öffentliche Beschreibung eines Web Services in XML), dadurch automatische Erzeugung eines Objekt-Proxys auf jeder Plattform möglich
- Zentrale Registrierung und Suche nach Web Services im UDDI (Universal Discovery Description and Integration; globale Datenbank über Unternehmungen) möglich
- Verschlüsselungs- und Authentifizierungsfeatures (tlw. noch in Arbeit) [Microsoft 2002e]

In Kapitel 4 wird das Web Service "miiCurrencyConverterService", welches im Rahmen der Diplomarbeit entwickelt wurde, vorgestellt. Dabei werden technische Aspekte von Web Services näher erläutert.

3.2.3. Eignung für ASP

Aus Sicht eines ASP-Kunden ist eine mit .NET entwickelte Applikation ein webbasiertes System, mit allen seinen Vor- und Nachteilen. In .NET wurden aber zahlreiche gute Ideen

umgesetzt, die die Entwicklung von standardisierten, webbasierten Komponenten vorantreiben könnte. Zur Zeit lässt sich die Leistungsfähigkeit von .NET nur schwer in Zahlen ausdrücken, da das System noch weiterentwickelt wird. Es wird aber nach einer Bewertung der neuen Konzepte (Web Forms und Web Services) doch der Versuch dazu unternommen.

Die Web Forms sind, für sich betrachtet, ein Fortschritt für die User Interface Designer und, durch die Trennung von GUI und Programmlogik und die "Software-Reuse"-Möglichkeiten, auch für die Programmierer. Design, Implementierung und Wartung wird vereinfacht und beschleunigt. Soweit es die Beurteilung der aktuellen .NET-Version zulässt, gibt es aus Sicht eines Anwenders, in Bezug auf die benutzten Protokolle (HTTP) und Client-Technologien (HTML, JavaScript), keine Fortschritte zu bisherigen Webtechnologien. Serverseitig sieht es freilich anders aus: Besonders die automatische Anpassung an den verwendeten Browser kommt dem ASP-Modell sehr zugute.

Möglicherweise lässt sich mit .NET die Funktionalität und Produktivität von webbasierten Applikationen steigern. Ob dies aber wirklich der Fall sein wird, wird sich zeigen sobald genügend Applikationen am Markt sind.

Zweifellos sind die Web Services eine sehr gute Idee und ein Fortschritt in der standardisierten Kommunikation im Internet. Beim erfolgreichen Einsatz der Web Services sind im ASP-Bereich verschiedene Szenarios denkbar:

1. **Web Services ohne Integration:** Es ist eher unwahrscheinlich, dass die Web Services alleine, für sich isoliert, für den Endanwender interessant werden. Web Services bieten außer der im letzten Kapitel angesprochenen "Testseite" keine Benutzerschnittstelle und daher keine Interaktivität. Web Services bieten nur "Business Logic", also Funktionalität in Form von Methoden (die Prozeduren oder Funktionen sein können). Im Rahmen der Diplomarbeit ist ein solcher "ASP Dienst ohne grafische Benutzerschnittstelle" entstanden und wird im Kapitel 4 vorgestellt.

2. **Web Services-Integration beim ASP:** Die Integration von Web Services, unter Zuhilfenahme von Web Forms und zusätzlicher Applikationsprogrammlogik, zu einer Applikation kann aber (serverseitig) beim ASP erfolgen. Sinn macht dies insbesondere, wenn sich die Applikation an Funktionalität und Informationen (Daten) aus dem Internet bedient oder bedienen muss. Auch in dem Sinne des "Software-Reuse" trägt dies zur effizienteren Softwareherstellung bei und wird deshalb immer intensiver betrieben. Beispiele sind: Börsenverwaltungssoftware, die Börsenkurse per Web Services lädt, Web Shop-Lösungen, die Daten von einem Datenbankserver beziehen, der nicht beim ASP steht, oder Suchmaschinen (für Lagerbestandsabfragen u.ä.). Alle diese Applikationen stehen dem Anwender webbasiert zur Verfügung.

   Bei monolithischen Applikationen hingegen, die selbst keine Internetanbindung zu anderen Servern benötigen, sind Web Services nicht unbedingt erforderlich oder sinnvoll.

3. **Web Service-Integration beim Anwender**: Ebenfalls denkbar ist, dass Firmen Web Services mieten werden, um sie softwaretechnisch z.B. in ihre Geschäftsprozesse zu integrieren. Hierfür erfolgt die Integration der Dienste auf Clientseite. Deshalb ist nicht auszuschließen, dass in Zukunft Web Services gegen Gebühr vermietet werden, also als ASP-Lösungen angeboten werden. In diesem Fall wird die eigentliche Applikation nicht gemietet, sondern am Client installiert. Es ist dabei egal, von welchem Hersteller (vom Anwender, vom Web Service-Provider oder von einem Drittanbieter) die lokale Software stammt, da der Zugriff auf Web Services ja standardisiert ist. In diesem Zusammenhang sei nochmals der "MSN Explorer", der sich in Zukunft Web Services von Microsoft bedienen wird, zitiert [MSN 2002].

4. **Web Service-Integration bei Anbieter und Anwender**: Eine vierte Möglichkeit ist, dass es keinen dezidierten Client und Server gibt. Mehrere Unternehmungen greifen gegenseitig auf die angebotenen Web Services zu ("E-Collaboration"). Da die Veröffentlichung (das "Deployment") eines Web Services technisch und organisatorisch viel einfacher zu bewerkstelligen ist, als das Anbieten von Applikationsgesamtlösungen über Internet, ist diese Variante vor allem für E-Business-Anwendungen im B2B-Rahmen sinnvoll.

Für "reines ASP", bei dem also die Benutzung der Anwendung eine zentrale Rolle spielt, erscheint nur Variante 2 geeignet, da die Benutzerschnittstelle ein integraler Bestandteil der meisten Applikationen ist. Es sollte keine applikationsspezifische GUI- oder gar Applikationskomponente, eventuell sogar von der Softwaremiete komplett losgelöst, auf dem Rechner des Clients installiert werden müssen, da dadurch eine Reihe von Vorteilen von ASP wieder zunichte gemacht werden würde.

Zum Abschluss eine Diskussion der technischen ASP-Aspekte, die im theoretischen Teil behandelt wurden:

Als Entwicklungswerkzeug und Plattform für webbasierte Applikationen, bietet es eine gute Anbindung an **Datenbanken**: Web Forms können ihre Attribute statisch oder dynamisch aus einer Datenbank beziehen. Über eine Zwischenstufe, den sog. "DataSets", eine Art Datenbank-Cache, lässt sich diese Anbindung erreichen, die auch über Web Services möglich ist.
Die **Dateisysteme** (von Client und Server) werden, wie bei allen webbasierten Systemen, nur unzureichend unterstützt. Es gibt weder serverseitig (spezielle Datenstrukturen, etc.) noch clientseitig (Dateiauswahlboxen etc.) entsprechende Unterstützung.

Die **Sessionverwaltung** ist gut gelöst. Web Forms speichern auf Wunsch mittels Servervariablen oder Cookies automatisch ihren Sessionzustand. In dem "Session"-Objekt ist immer der aktuelle Zustand pro Benutzer gespeichert. Sessions lassen sich auch persistent in eine Datenbank übertragen, sodass ein hoher Grad an **Personalisierung** erreicht wird.

Die **Kommunikation** läuft beim Browser über HTTP ab (es ist aber auch SSL-Tunneling geplant), mit anderen Systemen über XML und SOAP (auch hier liegt HTTP zugrunde).
Die **Mehrbenutzerfähigkeit** ist bei .NET gut ausgeprägt. Der Middle-Tier, COM+, ist verantwortlich für die Verwaltung mehrerer Sessions gleichzeitig. COM+ verwaltet die gesamte Applikationsprogrammlogik.

Die **Applikationsprogrammlogik** ist bei .NET folgendermaßen angelegt: Die Application-Klasse erzeugt für jeden Online-Benutzer ein Session-Objekt und für jede Internetseite ein "Page"-Objekt. Dieses Objekt trägt den Code für die Business-Logik in einer .aspx-Datei. Dieser Code ist ereignisbasiert: Beim ersten Zugriff auf die damit assoziierte Internetseite wird der Constructor aufgerufen, der alle Web Forms initialisiert. Infolge wird ein "Page_Load"-Ereignis ausgelöst, das ASP.NET veranlasst, entsprechenden Code (HTML, JavaScript) an den Clientbrowser zu senden. Jeder vom Web Form unterstützte Zustandwechsel hat ein "Page_Load"-Ereignis zufolge, das die Controls automatisch neu rendert. Dies ist möglich durch die Speicherung von "Page"-Information (in die URL oder in Cookie- oder Servervariablen codiert). Es lassen sich auch eigene Page- oder Sessionvariablen definieren.

Die Seitenorientierung hebt .NET fast vollkommen auf: Eine Änderung der GUI kann direkt in den entsprechenden Event-Handlern der GUI-Elemente geschehen. .NET aktualisiert automatisch die HTML-Benutzerschnittstelle. Beim Wechsel (clientseitig z.B. mittels eines Links) auf eine andere .aspx-Datei bleibt zwar das Sessionobjekt vorhanden, aber das vorige Page-Objekt geht verloren und ein neues wird erzeugt. Mit Berücksichtigung dieser Einschränkung sind aber Umsetzungen von bisherigen Programmen, je nach Komplexität (der Applikation und dessen GUI) relativ einfach möglich.

Um die **Skalierbarkeit** und **Availability** zu verbessern, existiert ein eigener .NET Enterprise Server namens "Application Center Server" für den Zweck des Clustering (siehe Kapitel 2.6.2.2.). Im Vergleich mit J2EE-Servern wird [Paiha 2001] die bessere **Performance** hervorgehoben. Diese lässt sich für die Präsentationslogik noch vom "Internet Security & Acceleration Server" (ISA-Server) verbessern, der HTML Seiten zwischenspeichern kann. Die Voraussetzungen für **Geräteunabhängigkeit** sind wie bereits angesprochen, ausgezeichnet: Web Forms passen sich an die Fähigkeiten des Endgerätes an, ohne ein Plug-In zu benötigen, Web Services lassen sich von jeder Plattform, die HTTP unterstützt, benutzen und serverseitig lässt sich .NET in Zukunft durch die Sprachenunabhängigkeit auf viele Betriebsysteme portieren. Die **Integrationsmöglichkeiten** erscheinen durch die Web Services und die Verwendung des XML-Formats im Vergleich mit anderen webbasierten Plattformen, stark erweitert, wenn auch nur serverseitig. Es lassen sich COM+-Komponenten, herstellerunabhängige und sprachenneutrale Komponenten mit Applikationsprogammlogik, nahtlos integrieren. Auf dem Endgerät eröffnen sich diese Möglichkeiten derzeit nur bei der Installation von Spezialsoftware, jedoch nicht mittels eines Standard-Internetbrowsers. Die Programmierung der **Benutzerinterfaceprogrammlogik** wird durch ASP.NET wesentlich vereinfacht: Die Web Forms bieten Funktionalität auf einer wesentlich höherem Abstraktionsebene als HTML. Es gibt einen visuellen Designeditor. Die **Interaktivität** von .NET-Applikationen kann leider kein hohes Maß erreichen: Um die hohe Plattformunabhängigkeit zu bewahren, wird dem Browser nur HTML und JavaScript, oder WAP, bereitgestellt. Java oder ActiveX-Komponenten können zwar manuell integriert werden, die Features von ASP.NET unterstützen diese Technologien jedoch nicht. Die **Benutzerfreundlichkeit** von .NET Web-Applikationen ist in der aktuellen Version eingeschränkt. Die Auswahl an Web Forms ist sehr dürftig. Es sind beinahe nur Controls verfügbar, die mit HTML alleine auch darstellbar wären, d.h. JavaScript und andere Clienttechnologien werden nur von wenigen Controls eingesetzt. Zum Beispiel fehlt (noch ?) eine Control zur Darstellung eines grafischen Menus. Inwieweit die Entwicklung in dieser Richtung noch weitergehen wird, lässt sich nicht bestimmen.

Funktionalität und **Zuverlässigkeit** zu implementieren obliegt dem Programmierer.

Fazit:

.NET enthält eine Reihe von fortschrittlichen Techniken, die Applikationsdesignern, Implementierern und Anwendern von webbasierten Diensten die Arbeit erleichtern. Die Geräte-, Plattform- und Sprachenunabhängigkeit und die Browsererkennung von .NET könnte dem ASP Bereich einen neuen Schub vermitteln. Die Trennung von Benutzerinterfaceprogrammlogik und Applikationsprogrammlogik ist beispielhaft. Teilweise wird die Seitenorientierung von HTML aufgehoben, sodass relativ einfach Umsetzungen von herkömmlichen Applikationen mit grafischen Front-Ends möglich sind.

Web Services erscheinen u.U. für ASP geeignet. Microsoft wird selbst als ASP auftreten und die Produktpalette "My Services" [Microsoft 2002d] anbieten.

Für interaktive Applikationen mit Benutzeroberfläche ist .NET als webbasiertes System, mit allen besprochenen Vor- und Nachteilen, zu werten. .NET wurde aber nicht ausschließlich für ASP entworfen. Vielmehr erscheint es besonders für bestimmte Anwendungen des E-Business und der E-Collaboration interessant, die nicht notwendigerweise als Mietsoftware realisiert sein müssen.

3.3. BEA WebLogic und IBM WebSphere

Zwei prominente Serverplattformen, die die Entwicklung und den Betrieb von webbasierten Applikationen erleichtern, sind BEA WebLogic und IBM WebSphere. Wie sich noch herausstellen wird, sind auch diese Produkte (wie Microsoft .NET) eigentlich für Anwendungen des E-Business und der E-Collaboration konzipiert, der Fokus dieser Plattformen liegt daher in der Bereitstellung und dem Austausch von Geschäftsinformationen und Diensten, und nicht in der Vermietung und Bereitstellung von Applikationen, über das Internet.

BEA WebLogic und IBM WebSphere sind Java Application Server, die den J2EE (Java 2 Enterprise Edition)-Standard implementieren.

Auf [JDJ 2001] läuft (zum Zeitpunkt der Erstellung dieser Diplomarbeit) eine Umfrage unter Internetbenutzern, welches der beste Java-Applikationsserver sei:

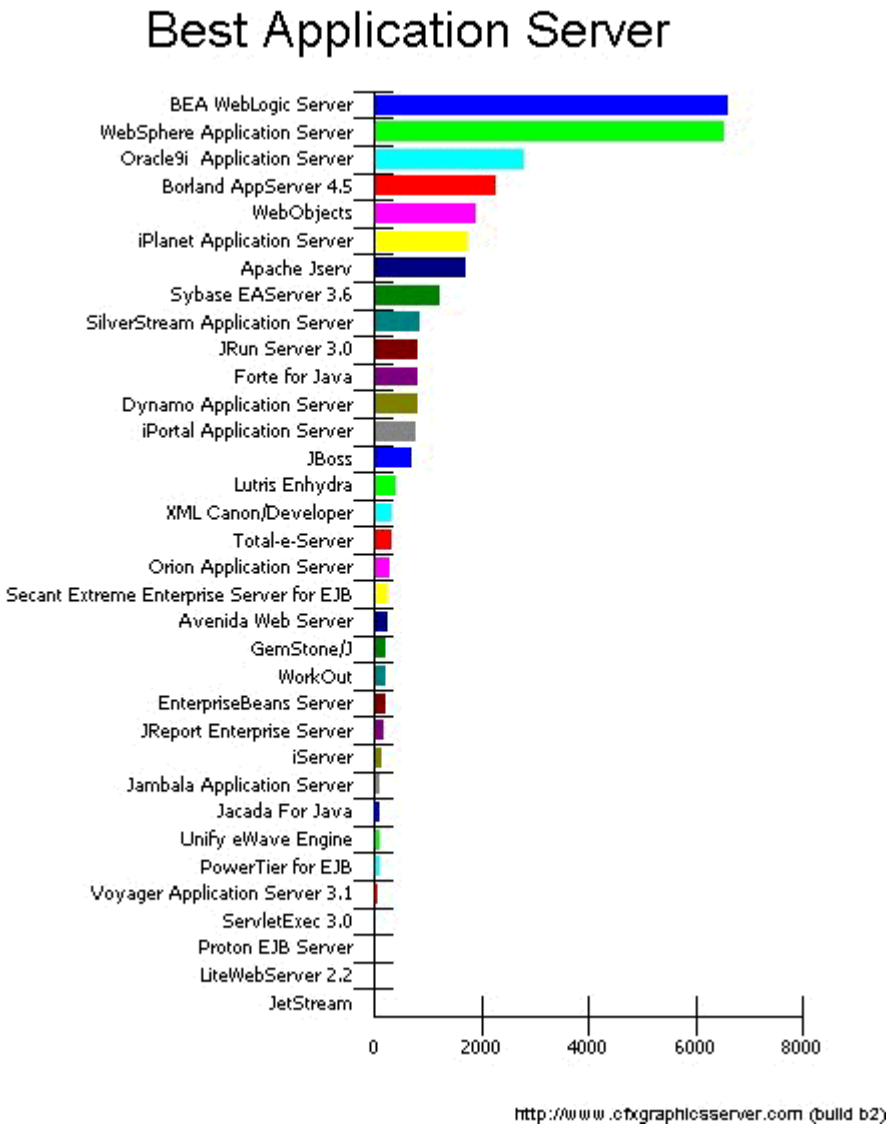

Abbildung 3-4 : Die besten Applikationsserver [JDJ 2001]

Mit jeweils 6610 und 6515 Stimmen liegen "BEA WebLogic Server" von BEA Systems und "WebSphere Application Server" von IBM weit vor "Oracle9i Application Server" von Oracle (2792 Stimmen) und Borland AppServer 4.5 von Borland (2270 Stimmen) (Stand: 18.1.

2002). Aus diesem Grund erscheint eine nähere Betrachtung dieser beiden Java Application Server interessant.

Von der Funktionalität her sind beide Applikationsserver aufgrund des gleichen J2EE-Kerns sehr ähnlich. Bei der Frage, ob BEA WebLogic oder IBM WebSphere der bessere Application Server sei, konzentrieren sich daher die Diskussionen hauptsächlich auf Kosten- und Performanceunterschiede. Die produktabhängigen Details und Unterschiede werden nur kurz gestreift, da die Eignung für ASP durch die neben dem J2EE-Standard zusätzlich implementierten Erweiterungen nicht wesentlich beeinflusst wird.

3.3.1. Java 2, Enterprise Edition

J2EE wurde entwickelt, um den Anforderungen des E-Business, wie Bereitstellung von einfach zu nutzenden Diensten für Kunden, Partner, Mitarbeiter und Lieferanten, gerecht zu werden.
J2EE stellt eine Spezifikation (ein Konzept) zur Entwicklung von Lösungen, die die Erstellung, den Einsatz und die Verwaltung von verteilten "multi-tier" Applikationen auf Java-Basis ermöglichen, dar.

Die wichtigsten Ziele von J2EE sind:

- **Vereinfachung der Entwicklung von Webapplikationen**

  J2EE-"Container" ermöglichen die Trennung von Applikationslogik (im E-Business: "Geschäftslogik") und Verwaltungsfunktionen, wie verteilte Kommunikation, Threading, Skalierbarkeit, Transaktionsmanagement, Ressourcenmanagement und Sicherheit. Diese Funktionen sind in dem sog. "Enterprise Java Beans (EJB)"-Container bereits enthalten und müssen daher nicht mehr implementiert werden. Programmierer können sich auf die Entwicklung von Applikationsprogrammlogik und Benutzerinterfaceprogrammlogik konzentrieren. Komponenten oder deren Funktionalität können über das Internet bezogen werden ("Software Reuse"). Das bedeutet schnellere, einfachere, kostengünstigere und möglicherweise daher bessere Lösungen.

- **Freie Wahl der Applikationsserver- und Betriebsystemplattform**

  J2EE-ensprechende Programme sind theoretisch auf jedem Applikationsserver lauffähig, der die J2EE-Spezifikation umsetzt, und auf jedem Betriebsystem, das die erforderlichen Java-Technologien unterstützt. Dies bedeutet serverseitige Plattformunabhängigkeit (nicht jedoch Sprachenunabhängigkeit). Application Server Hersteller fügen jedoch Zusatzfunktionalität (außerhalb der J2EE Spezifikation) zu den Serverprodukten hinzu, sodass Umprogrammierungen bei Migration notwendig werden.
  Die zwei wichtigsten Teile der J2EE-Spezifikation formen zusammen eine *Middleware* (siehe Kapitel 2.2.2.):

3.3.1.1. J2EE Plattform

Eine J2EE Plattform ist definiert als eine Standardplattform für den Betrieb von J2EE Applikationen, spezifiziert als eine Menge von APIs und Regeln. Diese Regeln bestimmen, auf welche Art eine nach der J2EE-Plattformspezifikation gerechte Umsetzung Webapplikationen zum Einsatz bringen muss und welche Internet-, Java-, und CORBA-Standards verwendet werden müssen. J2EE Umsetzungen der Plattform-Spezifikation werden "J2EE Server" oder "Application Server" genannt. Es gibt mittlererweile über 30 solcher Produkte am Markt. Zwei davon, "BEA WebLogic" und "IBM WebSphere" werden in dieser Diplomarbeit kurz vorgestellt.

3.3.1.2. J2EE "Application Programming Model"

Das J2EE-Applikationsmodell ist ein standardisiertes, "multi-tier" Programmiermodell für die Entwicklung von komponentenbasierten, "thin-client" Applikationen.

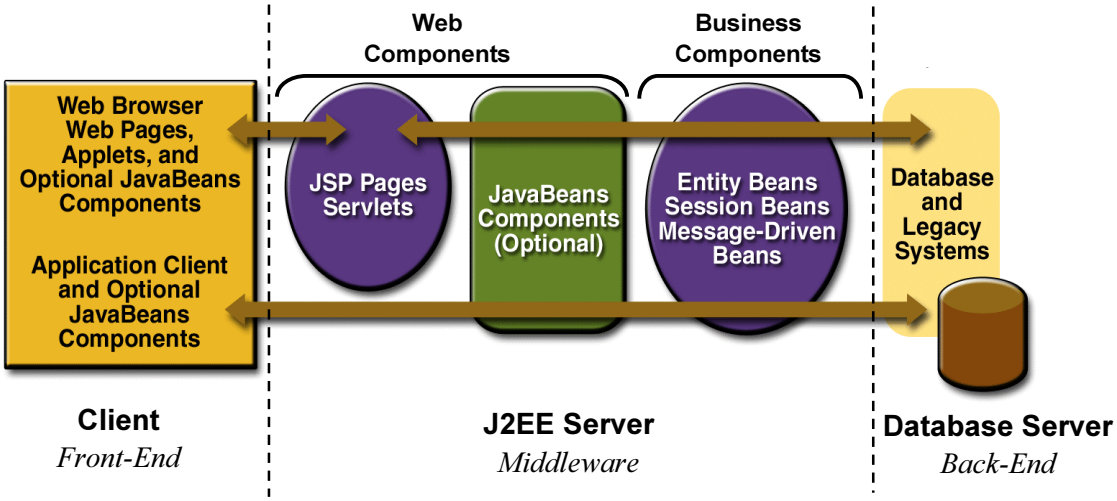

Abbildung 3-5 : J2EE Applikationsarchitektur

Die **Middleware** besteht im wesentlichen aus zwei Komponenten: Web Components und Business Components.

- **Web Components**

  Diese Komponente enthält die Präsentationslogik (Benutzerinterfaceprogrammlogik). Web Components können Java Server Pages (JSP) oder Servlets sein. Servlets sind Java-Klassen, die Internet-Anforderungen entgegennehmen und dynamisch Antworten (HTML, XML, etc.) generieren. Java Server Pages sind textuelle Einbettungen in HTML-Code.

  Statische HTML Seiten und Applets werden mit den Web Components gespeichert, gelten jedoch nicht als Web Components im Sinne der J2EE-Spezifikation. In Applets können optional clientseitige JavaBeans Komponenten enthalten sein, die mit den Web Components via XML kommunizieren. Diese Möglichkeit erscheint für ASP besonders interessant.

  Optional können in den Web Components JavaBeans enthalten sein, die Benutzereingaben verarbeiten und an die Business Components weiterleiten.

- **Business Components**

  Enterprise Java Beans (EJB) bilden die Business Components. Sie heißen so, weil mit ihnen bei E-Business Anwendungen die Geschäftsprozesse (Bankgeschäfte, Finanzierung, Einzelhandel) abgewickelt werden. Für ASP ist wesentlich, dass mithilfe der EJBs die gesamte Applikationsprogrammlogik abgebildet werden kann.

  EJBs empfangen (optional über das RPC-Protokoll "RMI IIOP") von den Web Components Daten vom Client oder vom Datenbankserver (über JDBC), führen Berechnungen durch, und senden die so verarbeiteten Daten weiter an die jeweils entsprechende Schnittstelle.

  Es gibt drei verschiedene Arten von Enterprise Beans. Eine *Session Bean* repräsentiert Funktionalität für eine Benutzer-Session. Wenn der Benutzer die mit der Session Bean

assoziierte Transaktion beendet hat, ist die Session Bean samt ihren Daten nicht mehr vorhanden. Eine *Entity Bean* hingegen repräsentiert persistente Daten-wie z.B. Datenbankeinträge - und die darauf arbeitenden Funktionen. Wenn der Client beendet oder der Server heruntergefahren wird, wird automatisch die Sicherung über die Back-End-Anbindung sichergestellt. Eine *Message-driven Bean* ist ähnlich einer Session Bean, kommuniziert aber asynchron über das Java Message Service (JMS).

Zur Middleware gehören noch die **Containers** und **Connectors**, die von der Applikationsserverfunktionalität bereitgestellt werden und für low-level Verwaltungsfunktionen wie Transaktionsmanagement, Multithreading, Ressourcenmanagement, Authentifizierung, Skalierbarkeit und Sicherheit sowie für die Anbindung an die Components zuständig sind. Dies erlaubt die Umsetzung dieser komplexen Aufgaben zur Einsatzphase statt in der Implementierungsphase.

Als **Client-Front-End** kann entweder ein Web Browser oder ein eigenständiger Java-Client ("Application Client") fungieren. Letztere Variante ist im ASP-Bereich nur sinnvoll, wenn es sich um einen universellen (d.h. für alle Applikationen geeigneten) Java-Client handelt.

Fazit:

Der J2EE Standard ist teilweise für die Konzeption von ASP Basistechnologien (Application Server) und entsprechenden ASP-Applikationen geeignet. Vor allem für den ASP und die Programmierer ist mittels J2EE-kompatiblen Application Servern und Entwurfswerkzeugen eine schnellere, einfachere, kostengünstigere, plattformunabhängigere und leichter zu wartende Implementierung von websprachenbasierten ASP-Applikationen möglich, dies aber ausschließlich bei Verwendung der Java-Technologien und -Sprache. J2EE ist auf jedem Fall ein Fortschritt zu dem bisherigen, unstrukturierten Einsatz der traditionellen Websprachen.

J2EE ist aber eigentlich geschaffen worden, um die Geschäftsabläufe (E-Business, E-Collaboration, Portale, etc.) im eigenen Unternehmen zu verbessern, indem J2EE-Komponenten von IT-Experten implementiert werden, nicht jedoch explizit um das ASP-Softwaremietmodell zu unterstützen. Beim E-Business kann zwar ASP eine Rolle spielen, die zwei Bereiche sind jedoch keinesfalls gleichzusetzen. Dies soll nun kurz erklärt werden:

Ein typisches Szenario für E-Business mit J2EE ist die Einrichtung von **speziellen** Applikationen für **firmeneigene** Zwecke (Webauftritt, Web Shop, Banking, etc.) auf einem Server im Unternehmen oder bei einem Hoster. Solche Anwendungen brauchen eine robuste Serverarchitektur mit Datenbankanbindung. Einzelne Dienste können zwar gegen Gebühr vermietet werden, doch steht die internetbasierte Anwendung nicht im Vordergrund.

Beim ASP-Modell soll hingegen **allgemeine Anwendungsfunktionalität** für viele Benutzer (1:n-Beziehung) vermietet und abgerechnet werden. Der ASP zieht im Normalfall nur aus der Vermietung einen geschäftlichen Vorteil, nicht jedoch aus dem Resultat der Anwendung der Applikationen durch die Kunden. ASP-Endkunden hingegen stehen mit der Applikation in engem interaktiven Kontakt. Für ihre Daten brauchen sie ein entsprechendes Ablagesystem. Außer für E-Business- und ERP-Anwendungen stellt hier ein Dateisystem eine viel geeignetere Lösung als eine Datenbank dar.

Aus dieser Überlegung heraus ist es verständlich, dass es auch bei J2EE keine Anbindung an das Serverdateisystem gibt und auf Technologien zur Schaffung von leistungsfähigen Benutzeroberflächen nur wenig Wert gelegt wird. Leider sieht der J2EE-Standard keine Neuerungen in der Clienttechnologie vor, und schreibt auch keine explizite Unterstützung für Browsersprachen wie JavaScript, VBScript, ActiveX, Plug-Ins, etc. vor. Lediglich das Konzept der JavaBeans, die in einem Java-Applet eingebettet sind und via XML mit der Middleware kommunizieren, könnte für ASP interessant sein, es sind auf dieser Basis jedoch noch keine

ASP Angebote bekannt. Der Mehrwert von J2EE-konformen ASP-Lösungen für den ASP-Kunden könnte sich zwar in höherer Performance, Ausfallssicherheit, Stabilität und Sicherheit widerspiegeln, an den grundsätzlichen Einschränkungen von browserbasierten Applikationen ändert sich jedoch nichts.

Trotzdem vermieten bereits viele ASPs erfolgreich Anwendungen, die über J2EE-konforme Server bereitgestellt werden, allen voran universelle E-Business- und ERP-Anwendungen. Diese brauchen nur mehr von der In-House Lösung zu ASP adaptiert werden, da sie bereits webbasiert sind.

Um die Eignung für ASP voll bewerten zu können, ist jedoch noch ein Blick auf praktische Beispiele von Applikationsservern zu werfen, da diese neben der J2EE Spezifikation natürlich noch beliebige weitere Funktionalitäten bieten können. Sowohl bei BEA WebLogic als auch bei IBM WebSphere handelt es sich dabei einerseits um interne, für Programmierer relevante Erweiterungen, als auch für den Anwender sichtbare Features.

Die folgenden Angaben beruhen jeweils ausschließlich auf die vom Hersteller offiziell im Internet veröffentlichten Einführungen in das jeweilige Application Server Produkt. Sie sind nicht als vollständig zu betrachten.

3.3.2. BEA WebLogic Server 6.1

Die "Introduction in BEA WebLogic Server 6.1" [BEA 2001] gibt über folgende Features Auskunft:

- Volle Implementierung des J2EE 1.2 Standards (tlw. auch 1.3)
- Verfügbar für MS Windows NT, Windows 2000, Sun Solaris, HP-UX, IBM AIX, Linux (Red Hat, SuSE, Red Flag), IBM z/OS, IBM OS/390 und Compaq Tru64 Unix
- XML Parser
- WAP Unterstützung (Nokia WAP server)
- Benutzerregistrierung (Subscriber Management) über beliebige Verzeichnisdienste (JNDI, LDAP etc.)
- Web Services über SOAP und entsprechende Web Service-Client Unterstützung
- Entity Beans können von mehreren Benutzern gemeinsam genutzt werden
- Stateless und Stateful Session Beans
- WebLogic T3-Protokoll: Optimierte Java-zu-Java Kommunikation
- SSL Unterstützung (HTTPS und T3S)

3.3.3. IBM WebSphere Server 4.0

Genau wie BEA WebLogic implementiert auch IBM WebSphere zur Gänze den J2EE Standard Version 1.2. Die folgende Zusammenstellung beruht auf [IBM 2001]:

- Volle Implementierung des J2EE 1.2 Standards (tlw. auch 1.3)

- Verfügbar für MS Windows NT, Windows 2000, Sun Solaris, HP-UX, IBM AIX, Linux (Red Hat, SuSE, Red Flag), IBM OS/400, IBM z/OS und IBM OS/390
- XML Parser
- WAP Unterstützung (IBM WebSphere Everyplace Suite)
- Benutzerregistrierung (Subscriber Management) über standardisiertes LDAP-Verzeichnis
- Web Services über SOAP, UDDI und WSDL
- J2EE Back-End-Connectors für SAP, PeopleSoft, Oracle ERP Financials, J.D. Edwards und einige IBM Produkte
- Zwei Entwicklerplattformen: IBM WebSphere Studio Workbench und IBM Visual Age for Java
- JDBC Datenbankadapter für Microsoft SQL Server, Oracle und Informix
- ActiveX (COM) - Schnittstelle zu Java Klassen
- Internationalisierungsfeatures
- Lightweight-third-party authentication (LTPA) für Authentifizierung von EJB-Komponenten auf Methoden-Ebene
- SSL Unterstützung (HTTPS)
- Integration mit "Tivoli" Service Operation Management Software

3.3.4. Java Application Server Verwaltungsfunktionen

Die meisten Application Server bieten folgende Verwaltungsfunktionen (nicht Teil der J2EE-Spezifikation):

- **Server als Web Server**

  Application Server lassen sich als primäre Webserver für die Bereitstellung von HTML, XML, Java Server Pages, Servlets, Java Klassen, Applets, Grafiken, Multimedia Dateien und andere Typen von Dateien verwenden. Eine J2EE-Webapplikation läuft im Web Container eines Web Servers.

- **Virtual hosting**

  Diese Option erlaubt auf einem Web Server mehrere logische Domains einzurichten, also mehrere Web-Sites auf einem Server zu betreiben. Diese Domains verweisen alle auf dieselbe IP-Adresse des (geclusterten) Web Servers. Via Servernameninformation im HTTP-Header bleibt der Hostname aus Clientsicht bei mehrfachen Anforderungen konstant.

- **Verwendung eines Proxyservers**

  Ein Proxyserver leitet Client-Anforderungen an verschiedene Server weiter und ermöglicht so Load-Balancing. Um den Session-Zustand eines Benutzers zu erhalten, gibt es

Replikationsmechanismen. Diese sichern den jeweils aktuellen Sessionzustand in eine Datenbank.

- **Clustering**

  Ein Server Cluster ist eine Gruppe von Serverinstanzen, die zusammenarbeiten, um eine ausfallssichere und skalierbare Plattform zu bilden. Dem Client erscheint ein Cluster als ein einziger Server. Bei BEA WebLogic werden Cluster vom Proxy über IP Multicast angesprochen.

  Einfach ausgedrückt ist Clustering von Applikationsservern das Gegenteil von Virtual hosting. Dabei verweist ein Domainname auf mehrere IP-Adressen. Der Proxyserver wählt nach einem definierten Schema einen Server aus und retourniert dessen IP-Adresse.

  Rechner innerhalb eines Clusters kommunizieren untereinander, um Sessioninformation auszutauschen. Alle JSPs und Servlets und die meisten Enterprise JavaBeans können geclustert werden. Bei WebLogic können nur Schreib-Lese Entity Beans nicht geclustert werden.

- **Pooling und Caching**

  Die Idee hinter Ressourcen-Pooling ist, eine Ressource nicht bei Anforderung zur Verfügung zu stellen, sondern beim Start des Servers oder zu definierten Ereignissen (z.B. in Netzpausen) mehrere Instanzen der Ressource auf einmal anzulegen. IBM WebLogic kann z.B. das Directory Service (JNDI), die Java Virtual Maschine und JDBC-Datenbankzugriffe poolen und cachen. Durch Vorausladen von Information und Diensten wird die Performance gesteigert.

- **Authentifizierung und Autorisierung**

  Die Authentifizierung dient zur Feststellung der Identität des Benutzers. Diese geschieht bei BEA WebLogic standardmäßig mittels eines JNDI-Lookups, bei IBM WebSphere mittels eines LDAP-Lookups. Internetbrowser unterstützen die Authentifizierung durch Verlangen von Benutzername und Passwort. WebLogic bietet zusätzlich Plug-Ins für LDAP, UNIX login service, Windows NT domain und RDBMS (Datenbankbasierte Berechtigungsprüfung).

  Autorisierung nimmt Rücksicht auf Zugriffsberechtigungen. Dazu gibt es in BEA WebLogic die ACLs (access control list). Eine ACL ist eine Liste von Benutzern und Gruppen, die berechtigt sind ein bestimmtes Service zu nutzen.

- **Verschlüsselung**

  Application Server unterstützen durchwegs die Secure Sockets Layer (SSL) -Technologie (siehe Kapitel 2.6.5.4.). WebLogic und WebSphere können auch so konfiguriert werden, dass auch Clients ein Zertifikat vorweisen müssen (Wechselseitige Authentifizierung).

- **Server Management und Monitoring**

  Jeder Application Server bietet ein gewisses Maß an Operation Management. Beide untersuchten Application Server erlauben eine webbasierte Konfiguration ("Konsole") von Diensten und ihrer Sicherheit, die Einrichtung von Applikationen sowie die Laufzeitkontrolle von Diensten. Beide Application Server Plattformen benötigen zur Einrichtung von Operation Management nur einen Server (zentrales Management).

3.3.5. Eignung für ASP und Vergleich mit Microsoft.NET

Fast alle Schlüsse aus Kapitel 3.2. (Microsoft .NET) treffen auch auf J2EE Server zu. Die zentrale Technologie von J2EE, nämlich EJB, ist COM+ von .NET sehr ähnlich [Paiha 2001].

Ebenso wie bei .NET gilt, dass durch ein klares und komponentenorientiertes Design, Applikationen schneller am Markt gebracht werden können und besser wartbar sind. Die Unterstützung von mobilen Geräten (WAP) ist hervorzuheben. BEA WebLogic und IBM WebSphere, genauso wie .NET, bieten Web Services (siehe Kapitel 4), obwohl diese Möglichkeit in der J2EE Spezifikation nicht vorgesehen ist.

Das J2EE Konzept ist vollkommen auf Java ausgerichtet, Dadurch ist es für sehr viele Plattformen verfügbar. Aus diesem Grund können ISVs mit Software für J2EE-Basistechnologien einen größeren Kundenkreis erschließen.

Mit .NET jedoch ist man nicht an eine Programmiersprache gebunden, wohl aber an das Betriebsystem Microsoft Windows. Die Middleware ist nur von Microsoft erhältlich und ist in Microsoft Windows ab Version 2000 integriert. Bei J2EE-Plattformen hat man eine breite Auswahl an Entwicklungswerkzeugen, bei .NET gibt es derzeit nur "Visual Studio.NET". Allerdings hat Borland schon für die zweite Jahreshälfte 2002 eine eigene .NET Entwicklungsumgebung angekündigt [Borland 2002].

Die Ausrichtung auf das E-Business wird bei den Java Application Servern noch stärker als bei Microsoft .NET betont. Bei Microsoft hingegen wird .NET auch eine Rolle im Unterhaltungssektor und im persönlichen Gebrauch von Software spielen.

Im Unterschied zu Microsoft.NET können sich JSPs und Servlets nicht an die Möglichkeiten des Endgerätes anpassen. Damit bleiben die Schwierigkeiten (Browserunterschiede, Seitenorientiertheit, etc.) bei der Implementation von interaktiven Browserschnittstellen bestehen. Ein für ASP interessantes Konzept von via XML mit dem Server kommunizierenden Java-Applets wird noch nicht erfolgreich eingesetzt.

Es folgt wieder eine Diskussion der technischen Aspekte von ASP, bezogen auf BEA Weblogic und IBM WebSphere:
Als Entwicklungswerkzeug und Plattform für webbasierte Applikationen, bieten BEA WebLogic und IBM WebSphere eine gute Anbindung an **Datenbanken**.

Die **Dateisysteme** (von Client und Server) werden, wie bei allen webbasierten Systemen, nur unzureichend unterstützt. Es gibt weder serverseitig (spezielle Datenstrukturen, etc.) noch clientseitig (Dateiauswahlboxen etc.) entsprechende Unterstützung.

Die **Sessionverwaltung** ist mittels der EJBs sehr gut gelöst. Session Beans speichern automatisch ihren Sessionzustand, solange er benötigt wird. Entity Beans sichern diesen in einer Datenbank. Dadurch wird ein hoher Grad an **Personalisierung** erreicht. Die **Kommunikation** läuft beim Browser über HTTP, HTTPS oder XML ab, mit anderen Systemen über XML und SOAP (auch hier liegt HTTP zugrunde).

Die **Mehrbenutzerfähigkeit** ist gut ausgeprägt. Für die Speicherung und Verwaltung von benutzerbezogenen Daten gibt es Technologien wie EJBs, JDBC und Directory Services wie JNDI oder LDAP.

Die **Applikationsprogrammlogik** ist in den EJB-Komponenten enthalten und in den Kapiteln 3.3.1.2. und 2.6.1.2. dargestellt worden. Ebenso wie bei .NET handelt es sich um ein ereignisbasiertes Programmmodell, bei .NET werden alle Ereignisse aber in der Applikationsprogrammlogik abgebildet. Die **Benutzerinterfaceprogrammlogik** ist strikt seitenorientiert. Die Seitenorientiertheit wird in der Benutzerinterfaceprogrammlogik der Servlets und JSPs abgewickelt, die Programmierung der Benutzerinterfaceprogrammlogik wird durch den J2EE Standard nicht wesentlich vereinfacht. Außer HTML, XML und Java-Applets werden keine anderen Clienttechnologien explizit unterstützt. Java-Applets setzen sich jedoch

derzeit wegen der langen Downloadzeiten, der schwachen Performance und der schlechteren Wartbarkeit im ASP-Sektor nicht durch. Jedoch gibt es für die Application Server eine Reihe von Entwicklungswerkzeugen (z.B. „IBM VisualAge for Java"). Die **Interaktivität** von J2EE-Applikationen kann leider kein hohes Maß erreichen: Um die hohe Plattformunabhängigkeit zu bewahren, werden dem Browser in der Regel nur HTML oder Java-Applets bereitgestellt. JavaScript, ActiveX-Komponenten oder Plug-Ins können zwar manuell integriert werden, die Features von Java Application Servern unterstützen diese Technologien jedoch nicht.

Die **Skalierbarkeit**, **Verfügbarkeit** und **Sicherheit** wird gegenüber der Kombination aus herkömmlichen HTTP-Servern und traditionellen Websprachen wesentlich besser sichergestellt. Allerdings erreichen WebLogic und WebSphere nicht die **Performance** von .NET., wie die .NET Portierung der J2EE Referenzapplikation "Pet Store" eindrucksvoll beweist [Microsoft 2002b].

Die Voraussetzungen für **Geräteunabhängigkeit** sind sehr gut, geräteabhängige Unterschiede in der Präsentationslogik müssen aber explizit implementiert sein. JSPs können prinzipiell jede Art von Daten (z.B. XML) über HTTP an den Client senden. Für WAP gibt es einen eigenen Server. Web Services lassen sich von jeder Plattform, die HTTP unterstützt, benutzen und serverseitig lässt sich J2EE durch die hohe Verfügbarkeit von Java auf fast alle Betriebsysteme portieren. Die **Integrationsmöglichkeiten** erscheinen durch die Web Services und die Verwendung des XML-Formats im Vergleich mit anderen webbasierten Plattformen, stark erweitert, wenn auch nur serverseitig. Es lassen sich herstellerunabhängige EJB-Komponenten nahtlos integrieren. Auf dem Endgerät eröffnen sich diese Möglichkeiten derzeit nur bei der Installation von Spezialsoftware, jedoch nicht mittels eines Standard-Internetbrowsers.

Über die **Benutzerfreundlichkeit** kann keine definitive Aussage getroffen werden, da eine entsprechende Test-Umgebung nicht zur Verfügung stand. Jedoch kann vermutet werden, dass sich die Unterschiede zu anderen websprachenbasierten Systemen in Grenzen halten, da von J2EE-konformen Application Servern nur herkömmliche Clienttechnologien wie HTML und Java verwendet werden.

Funktionalität und **Zuverlässigkeit** zu implementieren obliegt dem Programmierer.

Fazit:

BEA WebLogic und IBM WebSphere enthalten eine Reihe von fortschrittlichen Techniken, die Applikationsdesignern, Implementierern und Anwendern von webbasierten Diensten die Arbeit erleichtern. Im Unterschied zu .NET werden die Fähigkeiten verschiedener Endgeräte aber nicht berücksichtigt. Da nur HTML und Java als Browsersprachen verwendet werden, ist die Plattformunabhängigkeit sehr groß.

Es ist aus Sicht der ASP Basistechnologien als webbasiertes System, mit allen besprochenen Vor- und Nachteilen, zu werten. BEA WebLogic und IBM WebSphere wurden aber nicht speziell für ASP entworfen. Vielmehr erscheinen diese Produkte besonders für bestimmte Anwendungen des E-Business und der E-Collaboration interessant, die nicht notwendigerweise als Mietsoftware realisiert sein müssen.

Der Gesamtanteil der ASP-Lösungen, die über Java Application Server angeboten werden, ist noch eher gering. Viele Anbieter setzen noch auf traditionelle Webtechnologien wie z.B. CGI.

Am verbreitetsten sind E-Business, E-Collaboration und ERP-Lösungen. Der Grund hierfür ist, dass diese Arten von Software schon vor der Einführung von ASP als webbasierte Software etabliert waren. Für das ASP Modell müssen sie nur mehr angepasst werden.

3.4. Microsoft Windows Terminal Services

In Kapitel 2.1.1. und im Laufe des gesamten theoretischen Teils dieser Diplomarbeit wurden das Prinzip, die Möglichkeiten und die Einschränkungen von Terminalsystemen aufgezeigt.
Mehrfach wurden bereits die Terminalserver-Produkte "Windows Terminal Services" (WTS) von Microsoft und "MetaFrame" von Citrix erwähnt. MetaFrame bietet alle Funktionen von WTS und einige mehr. Aus diesem Grund wird WTS zuvor diskutiert. Falls nicht anders vermerkt, wurden die Informationen in diesem Kapitel über [Microsoft 2002c] bezogen.

3.4.1. WTS Konzept

WTS wurde erstmals 1998 in Windows NT 4.0, Terminal Server Edition (TSE) integriert. Seit der Windows 2000 Server-Familie ist es fixer Bestandteil des Betriebsystems und heißt nun "Windows Terminal Services".

Neben dem WTS-Server besteht WTS noch aus folgenden Komponenten:

- **Mehrbenutzerfähiges Kernel**

  Im Rahmen eines gemeinsamen Entwicklungs- und Lizenzierungsprojektes wurde hierzu in Windows NT 4.0 die von Citrix entwickelte "MultiWin"-Technologie integriert, die den Windows-Serverbetriebsystemen seitdem (Windows 2000 Server) Mehrbenutzerfähigkeit (siehe Kapitel 2.3.1.) ermöglicht.

  Für andere Serverplattformen oder Microsoft Betriebsystemversionen ist WTS nicht vorgesehen.

- **Remote Desktop Protocol (RDP)**

  Eine Schlüsselrolle spielt das Netzwerkprotokoll RDP, das dem Client den Zugriff auf den serverseitigen Terminaldienst ermöglicht.
  RDP basiert auf dem International Telecommunications Union's (ITU) T.120 Standard, unterstützt in der Version 5.0 LAN, WAN und RAS Dial-Up Verbindungen über niedrige Bandbreiten, drei Ebenen der Verschlüsselung, Datenkomprimierung, Druckerumleitung Netzlokalisierung und Sessionverwaltung und ist über sog. "virtual channels" erweiterbar. RDP ist nur über TCP/IP nutzbar und verwendet den TCP/IP-Port 3389.

  RDP verwendet einen eigenen Grafiktreiber auf der Serverseite um die Bildschirmausgabe umzuleiten, aufzubereiten und als RDP-Pakete an den Client zu senden. Auf Clientseite werden die RDP Pakete interpretiert und entsprechende Win32 GDI API Aufrufe zur Visualisierung ausgelöst. Auf Client-Eingabeseite werden Maus- und Tastatureingaben auf den Server umgeleitet. RDP verwendet eigene virtuelle Maus- und Tastaturtreiber um die Maus- und Tastaturereignisse zu empfangen.

- **Terminal Services Client**

  Die Terminal Services Clients sind ausschließlich für Windows Betriebsysteme verfügbar. Die Systemanforderungen sind mindestens ein 486/66 Mikroprozessor, eine 16-Bit VGA Grafikkarte und ein TCP/IP-Stack.

  Sie sind als eigenständiges Programm für Windows 3.11 for Workgroups, Windows 9X, Me, XP und CE, sowie Windows NT, 2000 (nur Intel) erhältlich. Unter [Microsoft 2002c] lässt sich die 3,4MB große Installationsdatei kostenlos herunterladen. Leider ausschließlich für Windows 2000 ist eine ActiveX-Komponente (Microsoft Terminal Services Advanced

Client, "TSAC") für den MS Internet Explorer ab Version 4.0 verfügbar. Allen anderen Clientsystemen bleibt der Zugriff auf ASP-Anwendungen über einen Webbrowser verwehrt.

Die TSAC-Setupdatei ist unter [Microsoft 2002c] zu finden und benötigt nur 318kB. Genau wie beim TSE sind bei Windows NT 4.0 Terminal Server Edition und Windows 2000 die Terminalclients schon beim Betriebsystem dabei. Nicht-Microsoft-Betriebsysteme und -Internetbrowser werden nicht unterstützt.

- **Terminal Server Administrierungswerkzeuge**

  Diese bestehen aus Terminal Services License Manager, Terminal Services Client Creator, Terminal Services Client Konfiguration und Terminal Services Manager.
  Genaueres zu dem komplexen Thema "Terminal Server Lizenzierung" lässt sich unter [Microsoft 2002c] nachlesen.

3.4.2. Leistungsmerkmale von WTS und RDP 5.0

Microsoft assoziiert die entsprechende RDP-Version mit der Terminal-Server-Version. Die aktuelle RDP Version 5.0. bietet folgende Leistungsmerkmale:

- **Anzeige**

  WTS unterstützt Auflösungen bis 1024 x 768 Pixel und bis 24 Bit Farbtiefe.

- **Verschlüsselung**

  Jede Version von RDP benutzt den RSA-Verschlüsselungsalgorithmus "RC4", welcher auch von Protokollen wie z.B. SSL benutzt wird. Als Schlüsselgröße sind 56 oder 128 Bit wählbar.

- **Komprimierung**

  Standardmäßig ist bei allen Terminal Sessions Komprimierung aktiviert, da es nahezu keine Performanceeinbußen verursacht. Bei eingeschalteter Komprimierung verringert sich die Menge der zwischen Client und Server gesendeten Daten etwa um die Hälfte. Da WTS nicht Open-Source ist, kann keine Aussage über das Komprimierungsverfahren gemacht werden.

- **Caching**

  Der WTS Client speichert Bitmap- und Schriftartdaten im RAM und auf Festspeicher. Der Festspeichercache beträgt 10MB und bleibt auch über mehrere Sessions bestehen. Dadurch wird der Netzverkehr reduziert.

- **Sessionverwaltung**

  Benutzer können die RDP-Verbindung während des Betriebes ohne expliziten Log-Off trennen. Dies kann entweder manuell geschehen oder in Folge eines unerwarteten Abbruchs durch einen Netz- oder Clientausfall. Wenn sich der Benutzer vom selben oder von einem anderen Gerät wieder anmeldet, wird er automatisch wieder mit seiner Session verbunden, solange dies innerhalb eines vom Terminal Server-Administrator definierten Time-Outs geschieht.

- **Unterstützung der Zwischenablage**

  Benutzer können Text und Grafik zwischen WTS-Sessions und dem lokalen Rechner ausschneiden, kopieren und einfügen.

- **Virtual Channels**

  Das RDP-Protokoll sieht eine Möglichkeit für die Übertragung beliebiger Daten vor, die sich "Virtual Channels" nennt. Damit können neue Anwendungen entworfen werden oder bestehende Anwendungen erweitert werden, die jede Art von Kommunikation (z.B. XML) zwischen Client und Server verwenden können.

- **Session Abbildung und Fernbedienung**

  Helpdesk-Mitarbeiter können bei Bedarf eine Terminal-Session sehen und kontrollieren (Fernbedienung). Eine Terminalsession (z.B. vom Helpdesk oder von einem Benutzer) lässt sich auf mehrere Endgeräte gleichzeitig "pushen" (Session Abbildung). Dies ermöglicht Schulungen und aktive Hilfeleistungen von einer zentralen Stelle (z.B. vom ASP).

- **Druckerumleitung**

  Anwendungen, die auf dem Terminalserver laufen, können über einen am Clientgerät am LPT-Port angeschlossenen Drucker ausdrucken. Andere Anschlüsse, wie COM oder USB, werden noch nicht unterstützt.

Darüber hinaus existieren noch zwei Funktionen, die das Betriebsystem zur Verfügung stellt:

- **Load Balancing**

  In Windows 2000 Advanced Server und Datacenter Server wurde ein rudimentäres Verfahren für Lastverteilung integriert, das auf einer zyklischen Zuteilung der Server über DNS beruht (Round-Robin-Verfahren). Dieses Verfahren stellt bei einer ungewollten Unterbrechung der Verbindung nicht sicher, daß beim Wiederverbinden wieder der korrekte Server ausgewählt wird. Server eines Windows 2000 Advanced- oder Datacenter Server-Clusters müssen selbständig für die Synchronisierung von Sessioninformation sorgen.

- **Internationalisierung**

  In Windows 2000 können gleichzeitig für jeden Benutzer unterschiedliche Ländereinstellungen vorgenommen werden. Der ASP kann dem Endkunden ein Service bereitstellen (z.B. Icon in der Taskleiste) mithilfe dessen der Benutzer regionale Einstellungen (wie z.B. die Zeitzone, Sprache der Applikation) vornehmen kann.

3.4.3. WTS Server Verwaltungsfunktionen

Zur Verwaltung von Funktionen, Benutzern und Sessions existieren folgende Administrierungstools:

- **Terminal Services Manager**

  Dieses Administrierungswerkzeug erlaubt die Verwaltung von Sessions, Benutzern und Prozessen. Einige der Hauptaufgaben sind:

- Trennen einer Verbindung
- Fernbedienung oder Abbildung einer Session
- Prozessabbruch
- Verbindungsstatus anzeigen
- Benutzer- und Clientstatus anzeigen
- Benutzer- oder Systemprozesse anzeigen
- Eine Nachricht (Message-Box) an eine Benutzersession schicken

- **Terminal Services Configuration**

  Mit dem Terminal Services Konfigurationswerkzeug können Sessions angelegt, verändert und gelöscht werden. Dies sind einige der wichtigsten Funktionen:

  - Konfigurieren einer neuen Verbindung
  - Verwalten der Zugriffsrechte für eine Verbindung
  - Benutzer und Gruppen zu einer Zugriffsrechtsliste hinzufügen
  - Time-Out Einstellungen und Verbindungsabbruchoptionen konfigurieren

- **Terminal Services Licensing**

  Terminal Services Licencing erlaubt Systemadministratoren Client-Lizenzen zu installieren und zu verwalten. Genauere Informationen finden sich im "Windows 2000 Terminal Services Licencing White Paper".

Zum Abschluss eine Zusammenstellung von für ASP relevanten Verwaltungstechniken:

- **Clustering, Load Balancing und Verwendung eines Proxy-Servers**

  Generell sind Terminalserverprodukte durch die monolithische Applikationsarchitektur hinsichtlich Clustering und Load-Balancing problematisch. Das Produkt WTS bietet diesbezüglich keine Unterstützung. Clustering und Load-Balancing muss auf (Windows-) Betriebsystemebene implementiert werden, notfalls auch durch Spezialsoft- und -hardware eines Fremdherstellers.

- **Authentifizierung und Autorisierung**

  Die Identität eines Benutzers wird über WTS standardmäßig genau wie eine lokale Anmeldung, nämlich über Windows-Log-On abgewickelt. Die Log-On-Bildschirmmaske ist also in dieser Betriebsart der Beginn der terminalbasierten Interaktion pro Applikationssitzung. Bei browserbasiertem Zugriff über den TSAC ist nur diese Möglichkeit vorgesehen. Als Option ist auch eine webbasierte Anmeldung möglich. Auf Serverseite werden die Log-In-Informationen an Windows-Log-On weitergeleitet.

  Zur Konfiguration der Benutzerautorisierung für Applikationen, Verzeichnisse und Daten dient das Terminal Services Configuration-Utility.

- **Verschlüsselung**

  Wie bereist angesprochen, bietet WTS Verschlüsselung über RC4. Allerdings wird serverseitig geregelt, welcher Verschlüsselungsgrad eingestellt ist. Der ASP-Kunde muss sich über den Verschlüsselungsgrad jeder Session bewusst sein.

3.4.4. Eignung für ASP

Konzipiert wurde der Server-Teil von WTS für den Einsatz in großen Unternehmen, die jedoch eine eigene IT-Infrastruktur (Server, Intranet) besitzen. Beim Einsatz dieses Produktes im ASP Bereich müssen in Bezug auf eine große Benutzeranzahl, heterogene Endgeräte und Internet- statt LAN-Anbindung einige Restriktionen in Kauf genommen werden.
WTS erscheint für ASP Anwendungen mit niedrigen Anforderungen (von seiten des ASP und von seiten der ASP-Kunden) durchaus geeignet. Als Terminalsystem bietet es die von Windows gewohnt leistungsfähige Benutzerschnittstelle. Zusätzlich bietet es Möglichkeiten, die webbasierte Systeme nicht bieten können, wie z.B. eine ausgezeichnete Anbindung an ein zentrales Dateisystem (aus Sicht des Anwenders).

Dies wird jedoch durch einen hohen Aufwand, um **Skalierbarkeit** und **Hochverfügbarkeit** zu erreichen, erkauft. Für WTS gibt es keine Möglichkeit des Load-Balancing oder Clustering auf Applikationsebene. Das bedeutet, dass dem Benutzer beim Verbindungsaufbau für die gesamte Dauer seiner Session ein dezidierter Server zugewiesen wird. Fällt ein Server aus, verlieren alle auf diesem Server eingeloggten Benutzer ihre Session. Da WTS keine Hochverfügbarkeitslösung ist, müssen auf anderen Soft- oder Hardwareebenen Lösungen geschaffen werden, die sehr teuer sind. Eine Installation von neuen WTS-Versionen muss für jeden Server in der Server-Farm manuell vorgenommen werden.

Hinzu kommt die wesentlich schlechtere **Wartbarkeit** eines **Dateisystems** im Vergleich mit einer Datenbank [Paiha 2001]. Dies ist jedoch bei allen Terminalserverprodukten der Fall und diese Probleme zu lösen ist nicht Aufgabe von Terminalbasistechnologien. Kurz gesagt ist der ASP Betrieb nur für eine kleine Benutzeranzahl und niedrige Service Levels zu empfehlen.

Außerdem werden einige Ziele von ASP nicht verfolgt. Zum Beispiel gibt es keine **Geräte- und Plattformunabhängigkeit**, weder auf Server- oder Clientseite, noch in der Protokollarchitektur. Besonders störend ist, dass ein selbstinstallierender, browserbasierter Terminalclient nur für Internet Explorer auf Windows 2000 verfügbar ist, welches für Thin-Client-Geräte unnötigerweise zu viele Ressourcen verbraucht. Deshalb ist man gezwungen, für jedes Endgerät mit älterem Microsoft Betriebsystem (z.B. Windows 95/98/Me) separat den Terminal Client zu installieren und zu aktualisieren. Allerdings rüsten viele Thin-Client-Hersteller ihre Geräte bereits mit den entsprechenden WTS-Technologien aus. An mobilen Geräten werden nur solche mit Windows CE unterstützt.

Die **clientseitige Wartung** verschwindet auch nicht ganz: Da ein webbasierter Aufruf der Applikationen (außer mit dem TSAC) nicht möglich ist, muss der Benutzer bei Updates regelmäßig händisch seine Verbindungseinstellungen anpassen. Darüber wird er mittels Message-Boxen informiert, die WTS über den Terminalclient "pushen" kann.

Die **Performance** von RDP ist ein viel diskutiertes Thema [Nieh 2000]. Windows 2000 Server (RDP 5.0) ist signifikant performanter (geringere Netzlast) als Windows NT 4.0 TSE (RDP 4.0). Die Anforderungen an die Netzqualität sind bei allen Terminalsystemen höher als bei den webbasierten Systemen. An gebräuchlichen Internetverbindungen existieren (nach ihrer Geschwindigkeit gereiht) Telefonleitungen, ISDN, Fernsehkabel (z.B. "Chello"), DSL und T1. Eine mit einem lokalen Betriebsystem vergleichbare Antwortzeit wird etwa ab ISDN-Geschwindigkeit erreicht. Laut Hersteller Microsoft benötigt WTS bei intensiver Nutzung eine Bandbreite von 48kbps.

Für die **clientseitige Anpassung** gib es einige Einflussmöglichkeiten, vor allem auf die Performance. Durch Anpassen der Auflösung bzw. Farbtiefe und der Aktivierung bzw. Deaktivierung von Caching, Komprimierung, Desktop-Hintergrund, Animierung von Fenster- und Menudarstellungen und der Anzeige des Fensterinhaltes beim Ziehen lässt sich das Maß an zu übertragenden Daten verringern und somit die Performance bei niedrigen Bandbreiten erhöhen. Eine automatische Skalierung auf die Auflösung des Endgerätes (wichtig für mobile

Geräte) gibt es jedoch nicht. Dadurch ist WTS für mobile Anwendungen nur dann geeignet, wenn diese speziell für kleine Bildschirmgrößen entworfen wurden.

Die clientseitige **Integration** ist bei WTS leider nicht so ausgeprägt wie sie bei Terminalsystemen sein könnte (z.B. bei Citrix). Zwar gibt es eine Integration der Zwischenablage und des lokalen Druckers, eine Integration des lokalen Dateisystems gibt es jedoch nicht. Dies schränkt den Nutzen für stark kooperierende Unternehmen sehr ein. Der Datenaustausch muss dann auf einem anderen Weg erledigt werden. ("Virtual Channels" von RDP; WWW oder FTP).

Die **Sessionverwaltung** ist bei RDP sehr gut. Bei Verbindungsabbruch ist innerhalb eines definierten Zeitraums eine Wiederaufnahme der Session möglich. Im Gegensatz zu den Websprachen wird exakt der Zustand zum Zeitpunkt des Ausfalles erhalten.

Die **Kommunikation** funktioniert bei WTS nur über RDP über TCP/IP, wobei ein spezieller Port benutzt wird, was in Hinblick auf eine korrekte Firewallkonfiguration von Bedeutung ist. Ein besonderes Feature, die "Virtual Channels", erlaubt die Übertragung von beliebigen Inhalten (z.B XML, HTML, Binaries) über RDP.

Die **Personalisierungsmöglichkeiten** sind bei WTS wesentlich besser als bei den Websprachen. Sowohl der Terminalclient sowie der serverseitige Desktop als auch die Applikationen selbst erlauben reichhaltige individuelle Anpassungsmöglichkeiten.

Die **Applikationsprogrammlogik** der Anwendungen ist beliebig. Es werden alle auf der Win32-Plattform lauffähigen Anwendungen (in Form des Binarycodes) bereitgestellt. Über die genaue interne Architektur von WTS kann keine Aussage gemacht werden, da der Quellcode von WTS nicht frei verfügbar ist.

Ebenso ist die **Benutzerinterfaceprogrammlogik** beliebig. Sie ist in den Anwendungen integriert. Eine Trennung von Applikationsprogrammlogik und Benutzerinterface-programmlogik geschieht mittels WTS zur Laufzeit während des ASP Betriebes.

Die **Benutzerfreundlichkeit** ist bei WTS durch die Funktionalität des gesamten Windows-Desktops wesentlich höher als bei den Websprachen. Windowsprogramme stellen eine wesentlich bessere, interaktivere Benutzerschnittstelle als Webprogramme zur Verfügung. Zudem sind die in den Applikationen eingebauten Hilfesysteme leichter zu bedienen.

Funktionalität und **Zuverlässigkeit** zu implementieren obliegt den Softwareherstellern.

Fazit:

Im Vergleich zu den Websprachen stellt WTS eine geeignetere Möglichkeit für ASP dar. Dies jedoch nur aus Anwendersicht. Aus Sicht des ASP ist WTS im Hinblick auf die angestrebten Service Levels von allen ASP-Basistechnologien am schwierigsten und am teuersten zu warten. Das Produkt selbst ist jedoch sehr günstig, da in Microsoft Windows 2000 integriert. Es müssen nur die Benutzerlizenzen zusätzlich erworben werden.

WTS ist vollkommen an die Microsoft Windows Plattform gebunden. Dies könnte die Einsatzmöglichkeiten von WTS von Fall zu Fall einschränken.

Einige Anwendungen aus dem E-Business- und ERP-Bereich sind besser für websprachenbasierte Systeme geeignet, da diese Anwendungen eine stärkere Integration ins World Wide Web und/oder eine Anbindung an Datenbanken benötigen. Zudem sind webbasierte Dienste auch mit mobilen Geräten einfach nutzbar.

WTS stellt eine komfortable Möglichkeit für die Bereitstellung von hochinteraktiven Applikationen in homogenen Endgerätestrukturen bei **niedrigen Service Levels** dar. Dies prädestiniert es für wenig unternehmenskritische Anwendungen (z.B. Textverarbeitung). Für hohe Service Levels (Skalierbarkeit, Performance, Ausfallssicherheit, etc.) ist das Produkt "Citrix MetaFrame" besser geeignet.

3.5. Citrix MetaFrame XP

Citrix MetaFrame stellt die derzeit leistungsfähigste Terminaltechnologie dar. Da Application Service Providing sehr hohe Ansprüche an die Basistechnologie stellt, wird Citrix im ASP-Bereich sehr erfolgreich eingesetzt. Nach Schätzungen der Gartner Group werden im Jahr 2005 an die 80 Prozent der größten ASPs Citrix Software einsetzen. Bereits im Juni 2000 setzten über 90 ASPs weltweit Citrix ein.

Am 13. Februar 2001 stellte Citrix die neueste Terminalserverfamilie für Windows namens Citrix MetaFrame XP (Extended Platform) vor, wobei die Buchstaben "XP" in keinem Zusammenhang mit "Microsoft Windows XP" stehen. MetaFrame XP ist derzeit ausschließlich für Windows Betriebsysteme mit integrierten WTS erhältlich, diese sind derzeit Windows NT 4.0 Terminal Server Edition und Windows 2000 Server, eine Anpassung an UNIX-Betriebsysteme ist aufgrund der plattformunabhängigen Architektur aber möglich. Citrix macht jedoch zum jetzigen Zeitpunkt keine Aussagen über die Realisierung einer MetaFrame XP-Version für UNIX Betriebsysteme. Es existiert jedoch noch eine ältere MetaFrame-Version für UNIX-Systeme, die WTS nicht voraussetzt, da WTS für UNIX nicht verfügbar ist. Allerdings fehlen dieser Version sehr viele Möglichkeiten der XP-Version und setzen sie in etwa mit WTS für Windows gleich.

Citrix MetaFrame XP basiert zwar auf einer anderen Architektur als WTS, bietet jedoch alle Leistungsmerkmale von WTS und eine große Anzahl zusätzlicher Leistungen und Funktionen, die erst das Bild einer kompletten ASP Basistechnologie vervollständigen.

Die neue MetaFrame-Familie besteht aus drei jeweils für einen bestimmten Kundenkreis zugeschnittene Produkte:

	MetaFrame XPs	MetaFrame XPa	MetaFrame XPe
Einsatz-Szenario	Support heterogener Clients, Zugriff über Netzverbindungen mit niedriger Bandbreite	Anwendungsbereitstellung in größerem Umfang, Zugriff über das Web	Unternehmensweite Anwendungs-bereitstellung in leistungsstarken, skalierbaren Serverfarmen
Anwendungen pro Server	Bereitstellung einzelner Anwendungen	Bereitstellung mehrerer Anwendungen	Middleware-Plattform für Anwendungen jeglicher Art
Anzahl der Server	1-5	2-50	10–1000+
Prozessoren pro Server	Anzahl der maximal unterstützten Prozessoren ist abhängig vom Be-triebssystem	Anzahl der maximal unterstützten Prozessoren ist abhängig vom Be-triebssystem	Anzahl der maximal unterstützten Prozessoren ist abhängig vom Be-triebssystem
Gruppierung der Server	Einzelne Server	Einzelne Serverfarm	Mehrere Serverfarmen
Benutzer pro Server	20-30	30-60	40-100
Einsatz Umfeld	Einzelne Abteilung, Arbeitsgruppe	Mehrere Abteilungen, Zweigstellen	Unternehmensweiter Einsatz
Microsoft Betriebssystem	Optimiert für die Windows 2000-Serverfamilie *)	Optimiert für die Windows 2000-Serverfamilie *)	Optimiert für die Windows 2000-Serverfamilie *)

*) unterstützt auch Windows NT 4.0 Terminal Server Edition

Abbildung 3-6 : Citrix MetaFrame XP Produktfamilie [Citrix 2002]

Nur MetaFrame XPe hat jedoch die für den ASP Betrieb wichtigen Funktionen zur Server-Verwaltung wie Load-Balancing, Skalierbarkeit, Systemüberwachung und Netzwerkmanagement integriert. Aufgrund des ASP Konzeptes, Kostenvorteile durch eine möglichst große Anzahl an Kunden zu erreichen, erscheint diese Version für ASP besonders attraktiv. Selbstverständlich muss im Einzelfall abgewägt werden, ob ein Einsatz einer anderen Version mithilfe von Zusatzprodukten kostengünstiger ausfällt. Um jedoch das Potential von terminalbasierten ASP Basistechnologien aufzuzeigen, wurde die Version XPe für eine

genauere Untersuchung ausgewählt. Das Leistungsportfolio der anderen Mitglieder der MetaFrame XP-Familie ist bis auf die eben genannten Managementfunktionen gleich.

3.5.1. Citrix MetaFrame XPe Konzept

Citrix MetaFrame XPe erweitert WTS über die Independent Computing Architecture (ICA) von Citrix durch zusätzliche Funktionalität auf Client- und Serverseite in folgenden Schlüsselbereichen:

- **Heterogene Computing-Umgebungen**

  MetaFrame gewährleistet Windows-basierten Applikationen den Zugang zu fast allen Arten von Client-Hardware, Betriebsystem-Plattformen, Netzverbindungen und LAN-Protokollen. Daher können Unternehmen ihre vorhandene Infrastruktur beibehalten, aber dennoch im ganzen Unternehmen die neuesten 32-Bit-Windows-basierten Applikationen über den ASP einsetzen.

- **Umfangreiche Verwaltungsmöglichkeiten**

  ASPs werden von MetaFrames Management-Werkzeugen profitieren, die unter anderem eine verbesserte System-Skalierbarkeit und einfachere Unterstützung von mehreren Applikationen für Tausende von Anwendern bieten. Über die Citrix Program Neighborhood können Administratoren dem Anwender schnell und problemlos Zugang zu neuen oder aktualisierten Applikationen auf Server-Basis verschaffen, ohne sich dabei Gedanken über die Client-Konfiguration machen zu müssen. Mit dem Load Management Service lassen sich mehrere MetaFrame-Server zu einer "Server-Farm" zusammenschließen, um den Anforderungen eines wachsenden Kundenstammes zu begegnen. Citrix Systemüberwachung und -analyse sorgen für eine umfangreiche Protokollierung, Systemüberwachung und die Möglichkeit, detaillierte Fakturierungsberichte zu erstellen. Die automatische Softwareverteilung in Serverfarmen schließlich automatisiert den Prozess der Replikation von Applikationen auf Citrix-Server.

- **Nahtlose Desktop-Integration**

  MetaFrame bietet eine erweiterte Funktionalität und mehr Komfort für den Anwender, unter anderem den vollständigen Zugang zu allen lokalen System-Ressourcen, wie z.B. 16-Bit Audio, Videostreaming (in begrenztem Umfang), lokale Laufwerke, COM-Schnittstellen und die Bereitstellung einzelner Anwendungen anstatt des gesamten Desktops.

Citrix basiert außer dem MetaFrame-Server noch auf folgenden Einheiten:

- **Mehrbenutzerfähiges Kernel**

  MetaFrame XPe für Windows baut auf der derselben Citrix-eigenen "MultiWin"-Technologie wie WTS auf. Citrix hatte die Technologie in Windows NT4.0 TSE und Windows 2000 Server integriert.

  UNIX Betriebsysteme sind bereits mehrbenutzerfähig, aus diesem Grund musste für MetaFrame für UNIX kein Kernel-Update durchgeführt werden.

- **Independet Computing Architekture (ICA)**

  Das ICA-Protokoll wurde von Grund auf plattformunabhängig gestaltet. Das ICA-Protokoll ist inzwischen der de facto Industriestandard für die virtuelle Übertragung von beliebigen Anwendungen auf allen Arten von Endgeräten. ICA unterstützt LAN, WAN, direkte serielle Kommunikation, direkte Wahlverbindungen, RAS Dial-Up Verbindungen und die Möglichkeit nach Citrix-Servern zu suchen. Neben TCP/IP werden IPX, SPX und NetBEUI Transportprotokolle unterstützt. Ebenso wie bei RDP existieren Features wie drei Ebenen der Verschlüsselung, Datenkomprimierung, Druckerumleitung, Netzlokalisierung, Sessionverwaltung und "virtual channels". Im Vergleich zu RDP verringert sich die Menge an zu übertragenden Daten etwa um die Hälfte. Dies wird durch Technologien wie "SpeedScreen 3" und die Effizienz des ICA-Protokolls erreicht.

- **Citrix Client**

  Citrix Clientprogramme sind zu allen MetaFrame-Versionen kompatibel. Es wird jedoch empfohlen, stets aktuelle Versionen zu verwenden, damit alle Funktionen, die der MetaFrame-Server bietet, genutzt werden können. Clientprogramme sind für folgende Plattformen verfügbar: DOS, Windows 3.1/3.11 for Workgroups/95/98/XP/NT/2000/CE, Macintosh, UNIX (Solaris/Sparc, Solaris/x86, SunOS, DEC, HP/UX, IBM, SGI, SCO, Linux), Java JDK 1.0/1.1, Plug-In für Netscapebrowser sowie als ActiveX-Komponente für den Internet Explorer. Alle Clientprogramme werden auf der Citrx-Homepage zum kostenlosen Download angeboten. Die Citrix-Clients für die Webbrowser sind zwar sehr klein (ca. 500kb), Einstellungen und Anpassungen müssen jedoch umständlich über die Registry vorgenommen werden, oder es wird zusätzlich ein eigenständiger Client installiert. Handys werden aufgrund des Fehlens einer Laufzeitumgebung noch nicht unterstützt. Allerdings unterhält Citrix mit vielen Herstellern mobiler Geräte Abkommen, die das ICA-Protokoll in ihre Produkte integrieren.

  Die Systemanforderungen sind in etwa gleich wie bei WTS.

- **Citrix MetaFrame XPe Administrierungswerkzeuge**

  Diese bestehen aus Citrix Load Management, Resource Management (Zusammenfassung aller Terminal Server Administrierungswerkzeuge) und Installation Management.

3.5.2. Leistungsmerkmale von MetaFrame XPe und ICA

Citrix nennt aufgrund der Abwärtskompatibilität von ICA keine Versionsnummern in Bezug auf das Übertragungsprotokoll. Die folgende Zusammenstellung beruht auf Eigenschaften von Citrix MetaFrame XPe in Kombination mit den aktuellen Citrix-Clients.

Als erstes werden die Funktionen, die auch in ähnlicher Weise bei WTS vorhanden sind, dargelegt:

- **Anzeige**

  Citrix unterstützt Auflösungen bis 64.000 x 64.000 Pixel und Farbtiefen bis 24 Bit.

- **Verschlüsselung**

  Citrix nutzt den https-Port der Firewall für den Citrix ICA- und XML-Datenstrom, sowie 40-, 56- oder 128-Bit Verschlüsselung der Internet-Standard-SSL-Technologie.

- **Komprimierung**

  Standardmäßig ist bei allen Citrix Sessions Komprimierung aktiviert, da es nahezu keine Performanceeinbußen auf Server- und Clientseite verursacht. Bei eingeschalteter Komprimierung verringert sich die Menge der zwischen Client und Server gesendeten Daten etwa um die Hälfte. Da Citrix MetaFrame nicht Open-Source ist, kann keine Aussage über das Komprimierungsverfahren gemacht werden. Allerdings lässt sich feststellen, dass ICA etwa um die Hälfte weniger Bandbreite als RDP benötigt. Laut Hersteller benötigt Citrix MetaFrame eine Mindestbandbreite von nur 20 kbps.

- **Caching**

  Der Citrix-Client speichert Bitmap- und Schriftartdaten im RAM und auf Festspeicher. Der Festspeichercache bleibt auch über mehrere Sessions bestehen. Dadurch wird der Netzverkehr reduziert. Die Größe des Cache lässt sich beliebig einstellen. Festlegen läst sich auch, ab welcher Größe Bitmaps auf dem Festspeicher gesichert werden.

- **Sessionverwaltung**

  Benutzer können die ICA-Verbindung während des Betriebes ohne expliziten Log-Off trennen. Dies kann entweder manuell geschehen oder in Folge eines unerwarteten Abbruchs durch einen Netz- oder Clientausfall. Sollte ein Netzausfall der Grund für die Unterbrechung sein, versucht sich der Client bei Reaktivierung des Netzes automatisch wieder mit dem Server zu verbinden. Wenn ein Absturz des Clients der Grund für die Unterbrechung war, kann sich der Benutzer vom selben oder von einem anderen Gerät wieder anmeldet. Er wird automatisch wieder mit seiner Session verbunden, solange dies innerhalb eines vom Terminal Server-Administrator definierten Time-Outs geschieht.

  Ein Citrix Client kann bis zu drei Master-Server definieren. Bei Verbindungsaufbau sendet der Client ein "Broadcast" an die festgelegten Master-Server. Derjenige Server, der als erstes antwortet, wird ausgewählt.

- **Unterstützung der Zwischenablage**

  Benutzer können Text und Grafik zwischen Citrix-Sessions und dem lokalen Rechner ausschneiden, kopieren und einfügen.

- **Virtual Channels**

  Das ICA-Protokoll sieht eine Möglichkeit für die Übertragung beliebiger Daten vor, die sich "Virtual Channels" nennt. Damit können neue Anwendungen entworfen werden oder bestehende Anwendungen erweitert werden, die jede Art von Kommunikation (z.B. XML) zwischen Client und Server verwenden können.

- **Session Abbildung und Fernbedienung**

  Helpdesk-Mitarbeiter können bei Bedarf eine Citrix-Session sehen und kontrollieren (Fernbedienung). Eine Terminalsession (z.B. vom Helpdesk oder von einem Benutzer) lässt sich auf mehrere Endgeräte gleichzeitig "pushen" (Session Abbildung). Dies ermöglicht Schulungen und aktive Hilfeleistungen von einer zentralen Stelle (z.B. vom ASP).

  Gespiegelte Sitzungen lassen sich über eine "Cancel"-Schaltfläche, die auf dem Benutzerbildschirm erscheint, jederzeit abbrechen.

Die folgenden Funktionen bietet MetaFrame zusätzlich zu WTS:

- **Drucker-, COM- und USB-Anschlussumleitung**

  Alle Geräte, die am LPT-, COM- und USB-Ports angeschlossen sind, werden unterstützt. Dies ermöglicht die Verwendung von einer Vielzahl an lokaler Peripherie wie Drucker, Scanner, Kameras etc. Der Citrix Universal Druckertreiber kann wahlweise eingesetzt werden, um die für Druckjobs zu übertragenden Datenmengen zu reduzieren.

- **Program Neighborhood**

  Diese sehr nützliche Funktion stellt einen "Dateimanager" für ASP Anwendungen dar. Bei Updates oder neuen Applikationen werden automatisch entsprechende Icons angezeigt. ASP-Applikationen können entweder webbasiert oder über die Program Neighborhood gestartet werden. Die Program Neighborhood ist auch der einzig direkt ausführbare (Executable) Teil des Citrix-Client Softwarepakets und Schaltstelle für Clientkonfigurationen.

- **Automatisches Client Update**

  Mithilfe dieses Features können ASP-Administratoren ein Update einiger oder aller registrierten Citrix-Clients veranlassen. Beim Start der Program Neighborhood werden die Benutzer über das Vorhandensein einer neuen Version informiert und brauchen nur dem Download mit darauffolgender vollautomatischer Installation zuzustimmen.

- **Video- und Audiounterstützung**

  Citrix unterstützt die Übertragung von Tonsignalen in wählbaren Qualitäten (8/16 Bit, Mono, Stereo, wählbare Abtastrate). Für die Audioübertragung kann eines von drei Komprimierungsverfahren ausgewählt werden. Eine Integration von Streaming-Video-Formaten ist in Arbeit.

- **Panning und Scaling**

  Bei Scaling können Benutzerauflösungen frei definiert werden. Eine voreingestellte Auflösung wird in diesem Fall mittels Interpolationsverfahren auf die benutzerdefinierte Auflösung reduziert oder vergrößert. Für spezielle Bildschirmauflösungen von mobilen Geräten stellt dies eine sehr nützliche Funktion dar.

  Beim Panning stellt Citrix automatisch Bildlaufleisten zur Verfügung, wenn die serverseitige Auflösung den Fensterbereich des Citrix-Clients überschreitet.

- **Seamless Windows**

  Unter Citrix muss nicht mehr der gesamte Remote-Desktop dargestellt werden. Hingegen können einzelne Anwendungen in eigenen Fenstern bereitgestellt werden. Diese Fenster verhalten sich wie lokale Tasks (z.B. Tastenkombinationen wie ALT-TAB). Zusätzlich lassen sich Verknüpfungen auf ASP-Anwendungen erstellen.

- **Einbindung von lokalen Laufwerken**

  Dies stellt eine große Stärke von Citrix dar. Ohne umständliche FTP-Programme können Daten sicher zwischen lokalen und ASP-Verzeichnissen ausgetauscht werden. Dazu wird

das lokale Laufwerk in die serverseitige, benutzerspezifische Umgebung gemappt. Laufwerksbuchstaben können frei gewählt werden.

- **Benutzerspezifische Zeitzonen**

  Im Gegensatz zu WTS übernimmt der MetaFrame XPe Server die clientseitg eingestellte Zeitzoneneinstellung.

- **Speedscreen 3 Latenzreduktion**

  Speedscreen stellt einen Sammelbegriff für Verfahren zur Überbrückung von Latenzzeiten und zur Verringerung der für die Bildschirmanzeige übertragenen Datenmengen dar. Derzeit sind zwei Funktionen zur Überbrückung von Verzögerungen implementiert:

  1. **Lokales Textecho** : Der Citrix-Client gibt eingegebenen Text lokal wieder. Dabei versucht er die verwendete Schriftart zu "erraten".
  2. **Mausklick Feedback** : Diese Option des ICA-Clients und des MetaFrame-Servers bietet ein beschleunigtes visuelles Feedback für Mausklicks. Benutzer sehen eine unmittelbare Reaktion auf ihre Eingaben.

- **Webbasierte Clientinstallation**

  MetaFrame XPe stellt eine Funktion ("NFuse") zum Download und zur automatischen Installation des korrekten Browserclients beim erstmaligen Aufruf einer ASP-Applikation über einen Webbrowser zur Verfügung.

- **Sonstiges**

  Mausradunterstützung, optionales Puffern von Maus- und Tastaturanschlägen (Verringerung der TCP/IP-Paketanzahl), Multi-Monitor Unterstützung, Übergabe von Parametern beim Start von ASP-Applikationen, Zuweisung von Betriebsystem-Tastenkombinationen (z.B. STRG-ESC) und clientseitige Ereignisprotokollierung.

3.5.3. MetaFrame XPe Server Verwaltungsfunktionen

Zur serverseitigen Verwaltung von Funktionen, Benutzern und Sessions existiert die Citrix Management Console, die folgende Funktionen beherbergt:

- **Resource Manager**

  Das Resource-Manager-Modul stellt eine zentrale Kontrolle von Servern, Anwendungen, Lizenzen, Druckern und Benutzern zur Verfügung. Er bietet alle Funktionen des Terminal Services Management- und des Terminal Services Configuration Utility und einige mehr:

  - Definierbare Messkriterien und Real-time Grafiken
  - Real-time Alarm über Bildschirm, Mobiltelefone und SNMP Traps bei Überschreiten von Schwellenwerten
  - Server Farm Monitoring: Wahlweise einzelne Server oder Server Farmen überwachen
  - Reporting
  - Programmierte Server Neustarts

- Netzwerkmanagementgrundfunktionen und Integration von Netzwerk-Management-Konsolen von Drittanbietern (HP OpenView, CA Unicenter, Tivoli NetView)
- CPU-Priorisierung
- Connection Control : Maximale Anzahl an Verbindungen festlegen
- Zentrales Lizenzmanagement
- Active Directory und Novell NDS Support : Veröffentlichung von Anwendungen

- **Load Management**

  Mit Citrix Load Management ist eine Lastverteilung pro Benutzer und Anwendung möglich. Eine Aufteilung des Betriebes einer Applikation auf mehrere Server ist jedoch im Unterschied zu manchen websprachenbasierten Systemen wegen des monolithischen Applikationsdesigns von Windows-Anwendungen nicht möglich.

  - Anwendungsbasierte oder serverbasierte Lastverteilung: Load Balancing Kriterien können pro Anwendung oder pro Server definiert werden
  - Load Balancing Regeln mithilfe zahlreicher Messkriterien definierbar (CPU Auslastung, Speicherbedarf, Ressourcenbedarf etc.)

- **Installation Management**

  Das Installation Management zentralisiert und vereinfacht die Anwendungsverteilung auf den Servern innerhalb einer Server-Farm.

  - Paketierung von Anwendungen, Dateien und Service Packs: Die zu verteilenden Anwendungen werden mitsamt den zugehörigen Installationsoptionen in ein Paket verwandelt, das auf die gesamten Server einer Farm repliziert werden kann. Dadurch wird sichergestellt, dass die Anwendung auf allen Servern identisch konfiguriert ist.
  - Automatisierte Installation und Deinstallation: Installationen und Deinstallationen können automatisiert zu Zeiten durchgeführt werden, die dem Administrator am günstigsten erscheint. Benötigt eine Installation einen Server Reboot, geschieht dies ebenfalls automatisch und etwaig eingeloggte Benutzer werden vorher informiert.
  - Server Gruppen: Durch die Definition logischer Servergruppen (z.B. Gruppe "Windows 2000 Server" oder Gruppe "Windows NT4") können Installationen gezielt auf den gewünschten Servern vorgenommen werden.
  - Inventarisierung : Alle installierten Anwendungen werden katalogisiert

Zum Abschluss eine Zusammenstellung von für ASP relevanten Verwaltungstechniken:

- **Clustering, Load Balancing und Verwendung eines Proxy-Servers**

  Generell sind Terminalserverprodukte durch die monolithische Applikationsarchitektur hinsichtlich Clustering und Load Balancing problematisch. Citrix MetaFrame XPe bietet Clustering und Load Balancing auf Anwendungsebene. Die Funktion eines Proxy-Servers übernimmt dazu ein speziell als "Master" konfigurierter MetaFrame Server.

- **Authentifizierung und Autorisierung**

  Die Identität eines Benutzers wird über Citrix standardmäßig genau wie eine lokale Anmeldung, nämlich über Windows-Log-On abgewickelt. Die Log-On-Bildschirmmaske ist

also in dieser Betriebsart der Beginn der terminalbasierten Interaktion pro Applikationssitzung. Als Option ist auch eine webbasierte Anmeldung möglich. Auf Serverseite werden die Log-In-Informationen an Windows-Log-On weitergeleitet.

Zur Konfiguration der Benutzerautorisierung für Applikationen, Verzeichnisse und Daten dient die Citrix Management Console.

- **Verschlüsselung**

  Wie bereist angesprochen, bietet MetaFrame Verschlüsselung über RC4. Der Kunde kann über Konfigurationsoptionen des Citrix-Clients frei wählen, welchen Verschlüsselungsgrad er standardmäßig oder für die aktuelle Applikationssitzung wünscht.

3.5.4. Eignung für ASP

Citrix MetaFrame XPe erscheint in jeder Hinsicht für ASP geeignet und wurde auch als einziges Produkt in Hinblick auf ASP designt. So ist MetaFrame XPe zusammen mit Windows 2000 Data Center Server aufgrund der ausgezeichneten **Skalierbarkeit** in der Lage, mehrere tausend Benutzer in einer entsprechenden Server Farm von Windows 2000 Servern zu unterstützen. Die **Performance** ist beim MetaFrame XPe etwa doppelt so hoch wie bei WTS. Dadurch wird sogar der Zugriff auf ASP-Anwendungen über ein 56k-Modem möglich. Ab einer gewissen Mindestbandbreite (20 kbps) ist die Performance des ICA-Protokolls im Unterschied zu RDP 5.0 weitgehend von der Netz-Bandbreite unabhängig.

Die **Kommunikation** funktioniert bei MetaFrame XPe über eine Reihe von routbaren und nicht-routbaren Protokollen wie TCP/IP, IPX, SPX und NetBEUI. Dabei können nahezu alle Verbindungsarten gewählt werden. Ein besonderes Feature, die "Virtual Channels", erlaubt die Übertragung von beliebigen Inhalten (z.B. XML, HTML, Binaries) über ICA.

Die **Sessionverwaltung** ist ähnlich gut wie bei WTS, nur noch etwas besser. Bei Netzunterbrechungen versucht sich der Client automatisch nach Reaktivierung des Netzes wieder zu verbinden. Der Client kann bis zu drei Master-Server definieren.

Im Vergleich zu WTS wurde die **Wartung auf Clientseite** nahezu vollständig eliminiert (automatische Clientaktualisierung, browserbasierte Clients), sowie die serverseitige Verwaltung wesentlich vereinfacht. Die automatische Installation von Service Packs, neuen Anwendungen etc. in Server Farmen ist eine unbedingt notwendige Einrichtung. Durch die integrierten Resource Management Funktionen ist eine Kontrolle der Service Levels auf direktem Weg möglich. Auf der anderen Seite ist die serverseitige Wartung trotzdem noch wesentlich umständlicher und teurer als bei den websprachenbasierten Systemen. Dies liegt zum einen an der schlechten Wartbarkeit von Dateisystemen (im Unterschied zu Datenbanken) und zum anderen an dem größeren Aufwand um **Ausfallssicherheit** und **Hochverfügbarkeit** zu erreichen, da Load Balancing und Redundanz nicht auf Modul- und Sessionebene funktionieren. Ein Benutzer ist vom Beginn bis zum Ende der Sitzung einem bestimmten Server zugewiesen. Der Einsatz von Redundanz und Hot-Standby-Komponenten für Terminalsysteme ist sehr teuer.

Das Produkt geht hervorragend mit der Situation von heterogenen Endgeräten bei der ASP Nutzung um. Citrix MetaFrame bietet auf Seiten des Endgerätes ausgezeichnete **Plattformunabhängigkeit**. Lediglich für manche mobile Geräte ist es aufgrund der kleinen Bildschirmgröße weniger geeignet. Auf Serverseite wird neben Windows auch UNIX unterstützt, leider aber nur von einer älteren, leistungsschwächeren Version. Es bleibt jedoch durch das plattformunabhängige Design von Citrix, im Unterschied zu WTS, die Hoffnung, dass eine MetaFrame XPe Version zukünftig auch für UNIX Systeme verfügbar sein wird.

Die clientseitige **Integration** wird bei Citrix durch eine Anbindung an das lokale Dateisystem sehr verbessert. Der dadurch ermöglichte Datenaustausch erlaubt (tlw. zusammen mit der Bildschirmspiegelung) sogar Anwendungen der E-Collaboration und erleichtert die

Datenmigration. Außerdem können nahezu alle lokalen Ressourcen wie z.B. Sound, Video, Drucker, COM-Anschlüsse und die Zwischenablage verwendet werden.

Für **clientseitige Anpassung** gibt es reichhaltige Möglichkeiten: Im Citrix Client gibt es Optionen für Auflösung, Bildskalierung, Klänge, Farbtiefe, Caching, Tastenkombinationen, Verschlüsselungsgrad, Latenzreduktion u.v.a.m; wesentlich mehr als bei WTS.

Die **Personalisierungsmöglichkeiten** sind bei Citrix wesentlich besser als bei den Websprachen, und noch etwas besser als bei WTS. Sowohl der Terminalclient sowie der serverseitige Desktop als auch die Applikationen selbst erlauben reichhaltige individuelle Anpassungsmöglichkeiten.

Die **Applikationsprogrammlogik** der Anwendungen ist beliebig. Es werden alle auf der Win32-Plattform lauffähigen Anwendungen (in Form des Binarycodes) bereitgestellt. Mit der UNIX-Version von MetaFrame können sogar UNIX-Applikationen zur Verfügung gestellt werden. Über die genaue interne Architektur von Citrix kann keine Aussage gemacht werden, da der Quellcode von Citrix nicht frei verfügbar ist.

Ebenso ist die **Benutzerinterfaceprogrammlogik** beliebig. Sie ist in den Anwendungen integriert. Eine Trennung von Applikationsprogrammlogik und Benutzerinterface-programmlogik geschieht durch MetaFrame XPe zur Laufzeit während des ASP Betriebes.

Die **Benutzerfreundlichkeit** ist bei MetaFrame durch die Funktionalität des gesamten Windows-Desktops wesentlich höher als bei den Websprachen. Windowsprogramme stellen eine wesentlich bessere, interaktivere Benutzerschnittstelle als Webprogramme zur Verfügung. Zudem sind die in den Applikationen eingebauten Hilfesysteme leichter zu bedienen. Im Vergleich zu WTS ist MetaFrame durch die Geräteunabhängigkeit und zahlreiche Zusatzfunktionen noch benutzerfreundlicher.

Funktionalität und **Zuverlässigkeit** zu implementieren obliegt den Softwareherstellern.

Fazit:

Citrix MetaFrame XPe ist das beste terminalbasierte System für ASP. Die meisten Applikationen können über Citrix MetaFrame XPe über das ASP Modell vertrieben werden. Die für ASP sehr wichtige Endgeräteunabhängigkeit wird von Citrix voll erfüllt. Lediglich für Handy-Applikationen ist Citrix nicht geeignet. Dass nur Windows Applikationen zur Verfügung gestellt werden können erscheint nicht als großer Nachteil, da nahezu jede Applikationsart für Windows verfügbar ist. Die Wartung wird nahezu vollständig vom Client zum ASP verlegt.

Das Management des ASP-Betriebs wurde durch die neue MetaFrame Version zwar vereinfacht, im Vergleich zu webbasiertem ASP ist die Anwendungsbereitstellung und -wartung jedoch wesentlich aufwendiger und teurer. Dies schlägt sich im Preis für ASP Angebote nieder. Deshalb werden webbasierte ASP-Dienste nach wie vor bestehen bleiben. Einige Anwendungen aus dem E-Business- und ERP-Bereich sind besser für websprachenbasierte Systeme geeignet, da diese Anwendungen eine stärkere Integration ins World Wide Web und/oder eine Anbindung an Datenbanken benötigen. Außerdem verfolgen neue websprachenbasierte Konzepte, wie J2EE, die Idee der Software-Wiederverwendung und -Modularisierung besser und erlauben so einen offenen Markt für Software-Komponenten. Zudem sind webbasierte Dienste auch mit mobilen Geräten einfach nutzbar.

Citrix MetaFrame XPe stellt die beste Möglichkeit für die Bereitstellung von hochinteraktiven Applikationen in heterogenen Endgerätestrukturen bei **hohen Service Levels** dar. Dies prädestiniert es auch für unternehmenskritische Anwendungen, die ohne ASP für KMUs nicht leistbar wären.

Die zahlreichen Vorteile von Citrix-basierten ASP-Angeboten schlagen sich jedoch zumeist in einem höheren Mietpreis nieder.

3.6. Proprietäre Lösungen

Unter diese Bezeichnung fallen alle ASP-Lösungen, auf die der Zugriff nur mittels einer speziell für die Applikation entworfenen, nicht standardisierten Clientsoftware möglich ist.
Dieses Kapitel bezieht sich also nur auf ein spezielles Clientsystem. Prinzipiell ist proprietäre Clientsoftware bei websprachenbasierten und bei terminalbasierten Server-Systemen einsetzbar und sogar Mischformen sind denkbar. Allerdings sind die Eigenschaften proprietärer Terminalsysteme den quasi-standardisierten Systemen wie Citrix und WTS recht ähnlich.
Proprietäre Clientsoftware wird bei ASP sehr selten angewendet, sie soll jedoch der Vollständigkeit halber erwähnt werden. Ein einheitliches Konzept gibt es bei proprietärer Clienttechnologie natürlich nicht.

SAP bietet z.B. sein ASP-Produkt "my.SAP" sowohl mittels Terminal-, Websprachen- (JAVA) als auch proprietärer Technologie an, um größtmögliche Flexibilität zu gewährleisten.

Daneben gibt es nichtkommerzielle und kommerzielle Angebote für den Massenmarkt wie z.B. MSN Explorer (kostenlos), ICQ (noch kostenlos) und Napster (kostenpflichtig), wobei sich die Hersteller aufgrund des Fehlens von SLAs nicht als ASPs bezeichnen.

Für ASP-Applikationen auf Basis von BEA WebLogic oder IBM WebSphere lassen sich Clients auf Basis von Java Swing-Klassen (Java Klassen für GUI-Programmierung) einsetzen.

Proprietäre Clienttechnologien haben den großen Vorteil, die **performanteste Benutzerschnittstelle** liefern zu können. Im Vergleich zu den webbasierten Systemen ist sie auch **leistungsfähiger** und **komfortabler**. Durch den Wegfall der Bindung an das HTTP-Protokoll ist man nicht mehr an die Seitenorientierung gebunden. Außerdem müssen Kontrolleingaben (siehe Kapitel 2.7.2.4.) nicht jedesmal zu einer Serverinteraktion führen. Dies erlaubt ein hohes Maß an **Interaktivität**. Allerdings müssen für die Darstellung eines kompletten virtuellen Desktops Terminaltechnologien eingesetzt werden, da sonst der clientseitige Aufwand zu hoch wird.

Für die **Kommunikation** lässt sich beispielsweise XML, RMI oder SOAP einsetzen. Der Bedarf an Bandbreite ist in der Regel niedriger als bei den Terminalsystemen, ob er auch niedriger als bei den Websprachen ist, hängt von der jeweiligen Anwendung ab.

Proprietäre Clientsysteme können sowohl Schnittstellen zum serverseitigen **Dateisystem** als auch zu einer **Datenbank** bieten. ASP-Angebote auf Basis von proprietären Clienttechnologien stellen eher einen Rückschritt dar, da wieder clientseitige **Wartung** anfällt. Für jede Art von Anwendung muß, im Unterschied zur Terminaltechnologie, ein eigenes Clientprogramm bezogen werden und installiert werden. Ein Clientsystem, das bei Änderungen an der Applikationsprogrammlogik kein Update benötigt, erfordert ein sehr restriktives Design, das viele Vorteile von eigenständigen Clientlösungen wieder zunichte macht. Daher ist man bei einem Applikationsupdate oft zu einem clientseitigen Update gezwungen. Die Größe der Installationsdatei für proprietäre Clientsysteme übersteigt in der Regel 4 MB, sodass man sich beim Einsatz eines Modems auf längere Downloadzeiten einstellen muss.

Die **Geräte- und Plattformunabhängigkeit** ist normalerweise sehr schwach ausgeprägt. Nur Clientsysteme auf Basis von Java erlauben ein gewisses Maß an Plattform- und Geräteunabhängigkeit, in der Praxis verwendet man jedoch aus Performancegründen meist Native-Clients, die nicht plattformunabhängig sind. Für mobile Geräte (PDAs, Handys) müssen in jedem Fall spezielle Front-Ends installiert werden. Proprietäre Clientsysteme bieten nicht die Möglichkeit, sich an das jeweilige Endgerät anpassen zu können.

Die **Benutzerfreundlichkeit**, **Anpassbarkeit** und **Personalisierungsmöglichkeiten** der Applikationen sind zumeist besser, da proprietäre Clientsysteme nicht von den Einschränkungen browserbasierter und terminalbasierter Systeme betroffen sind.

Für die Implementation der **Sessionverwaltung**, der **Benutzerinterfaceprogrammlogik**, von **Funktionalität**, **Integrationsmöglichkeiten** und **Zuverlässigkeit** gibt es keine allgemeinen

Kriterien. Alle anderen jetzt nicht genannten technischen Aspekte von ASP (Sicherheit, Hochverfügbarkeit, Applikationsprogrammlogik, Mehrbenutzerfähigkeit, Skalierbarkeit, Performance etc.) beziehen sich auf die Server-Komponente, die bei einem proprietären Clientsystem ein beliebiges, websprachenbasiertes oder terminalbasiertes System sein kann.

Fazit:

Proprietäre Clienttechnologien bieten zwar ein hohes Maß an Komfort, können jedoch zwei wichtige Ziele von ASP, nämlich Plattformunabhängigkeit und Wegfall von clientseitiger Wartung, nicht ganz erfüllen. Während die mangelhafte Plattformunabhängigkeit in vielen Fällen noch als wenig relevant eingestuft wird, wird die anfallende clientseitige Wartung von ASP Kunden kaum akzeptiert, denn die Verschiebung von Wartungsaufgaben an den ASP ist ja ein wichtiges Kriterium für die Wahl einer ASP-basierten Lösung. Bei Verwendung von ASP Basistechnologien, die einen automatischen Download und eine automatische Installation von Clientsoftware erlauben, wie z.B. "Java Webstart", können proprietäre Clienttechnologien jedoch in Einzelfällen eingesetzt werden.

3.7. Abschließende Ergebnisse

Vergleicht man die vorgestellten ASP Basistechnologien, so fällt auf, dass websprachenbasierte und terminalbasierte Technologien höchst unterschiedliche Ansprüche erfüllen.

Die Vergleichstabelle auf der nächsten Seite fasst die daraus gewonnen Erkenntnisse zusammen, wobei proprietäre Systeme aufgrund ihrer nicht generalisierbaren Eigenschaften ausgenommen wurden.

Deutlich zeigen sich die Vorteile von Citrix MetaFrame XPe. Nur bei Server Integration, Performance, Skalierbarkeit, Server-Wartbarkeit und der Unterstützung von mobilen Geräten wie z.B. Handys ist Citrix den Application Servern unterlegen. Grenzbereiche sind Hochverfügbarkeit, Mehrbenutzerfähigkeit und Performance. Diese Qualitäten können mit Citrix MetaFrame XPe nur mit etwas höherem Aufwand und höheren Kosten bewerkstelligt werden. Vergleichsweise schlecht schneiden traditionelle Internetsprachen gemeinsam mit herkömmlichen Web Servern ab. Ihre Bedeutung für ASP wird in nächster Zeit stetig abnehmen [Paiha 2001].

Kriterien	Websprachenbasierte Basistechnologien			Terminalbasierte Basistechnologien	
	Traditionelle Internet-sprachen	*Microsoft .NET*	*IBM WebSphere BEA WebLogic*	*Microsoft WTS*	*Citrix MetaFrame XPe*
Mehrbenutzer-fähigkeit	~	+	++	~	+
Endgeräte Plattform-unabhängigkeit	~	++	+	--	++
Endgeräte-Integration	--	~	~	~	+
Server-Integration	-	+	+	--	-
Benutzer-schnittstellen-leistungsfähigkeit	-	-	-	++	++
Performance	~	+	~	-	~
Skalierbarkeit	~	++	++	-	~
Hochverfügbarkeit	-	++	++	-	+
Unterstützung für dateiorientierte Applikationen	--	--	--	++	++
Unterstützung für datenbank-orientierte Applikationen	+	++	++	+	+
Unterstützung für mobile Endgeräte	~	++	+	--	-
Sessionverwaltung	~	+	+	+	++
Sicherheit	~	+	+	+	+
Endgeräte-Wartbarkeit	~	~	~	-	+
Server-Wartbarkeit	-	+	+	~	~
Endgeräte-Anpassung	~	+	~	~	~
Interaktivität	~	~	~	+	+

++ ... sehr gut
+ ... gut
~ ... befriedigend
- ... genügend
-- ... nicht genügend

Tabelle 8 : Eigenschaften von ASP Basistechnologien

Kapitel 4

ASP am Beispiel von "miiCurrencyConverterService"

Im Rahmen der Diplomarbeit wurde bei der Firma **mii-marcus izmir informationsmanagement ag** (Wien) ein einfaches **.NET Web Service** (siehe Kapitel 3.2.2.2.) namens **miiCurrencyConverterService** (im folgenden "**MCCS**" genannt) entworfen.

Die Funktion des Web Services ist die Umrechnung von Landeswährungsbeträgen auf Basis von täglich aktualisierten Kursen.

Dabei werden alle Euro-Währungen zuzüglich alle Währungen, die von der EZB (Europäische Zentralbank) als sog. "Euro Referenzwährungen" deklariert sind, berücksichtigt. Das sind Währungen, die sich zwar nicht in der europäischen Währungsunion befinden, für die Nationalbanken (in diesem Fall die Österreichische Nationalbank (ÖNB)) aber Kursempfehlungen (Umrechnungskurse zum Euro) an alle lokale Banken abgeben können. Es handelt sich dabei um einen Mittelwert für An- und Verkauf von Devisen in Österreich, da aber jede lokale Bank eigene Kurse stellen kann, ist die Genauigkeit der Umrechnungskurse für einfache Anwendungen gut genug.

Der Download der Kurse erfolgt täglich durch einen externen, serverseitigen Dienst von www2.oenb.at. Die Kurse, im Excel-Format, werden in ein textbasiertes Format umgewandelt und auf dem Serverdateisystem gespeichert.

Folgende allgemeine Merkmale von ASP (Kapitel 1.1.) treten zutage:

- **Zentrale Speicherung, Verwaltung und Echtzeitbereitstellung von Applikationen und Daten**

  MCCS speichert und verwaltet die Kurse (die Daten) zentral und stellt sie als Anwendungsfunktionalität in Echtzeit über das Internet bereit.

- **Webbasierter Zugriff**

  Der Zugriff erfolgt über das Internet per Browser (Web Service-Testseite), im einfachsten Fall ist keine Installation von Clientsoftware erforderlich.

- **Konzentration aller Beteiligten auf Kernkompetenzen**

  Der Anwender nutzt Applikationsfunktionalität von der MII, die für ihn durch die täglich aktuellen Kurse einen Mehrwert erhält. Die ÖNB stellt die Kurse bereit, und die MII stellt Funktionalität, die auf diesen Daten beruhen, bereit.

- **Minimale Anpassung an den Kunden**

  Es gibt keine personalisierte Anpassung.

Auf der anderen Seite treffen zwei ASP-Merkmale nicht zu:

- Es gibt keinen Mietvertrag und auch keine SLAs, da das Service kostenlos ist.
- Es gibt daher auch keine Verrechnung.

Die technische Auseinandersetzung von Aspekten des ASP-Softwarebetriebs erfolgt im Anschluss an die detaillierte Beschreibung des Web Services.

4.1. Funktionalität von MCCS

MCCS stellt folgende Funktionalität bereit:

- Währungsbeträge nach Tageskurs umrechnen
- Optionale Umrechnung zu einem bestimmten Kurs eines vergangenen Tages (zurück bis zur Freigabe des Web Services)
- Liste der unterstützten Währungskürzel (nach ISO-Standard, z.B. EUR, ATS, USD) abfragen
- Landeswährung über LocaleID (Gebietsschema-Code der Ländereinstellung) abfragen
- Datum der aktuellsten Kurse abfragen (alle Kurse werden gleichzeitig täglich zwischen 15:00 und 16:00 aktualisiert)

Das Web Service ist (zum Zeitpunkt der Diplomarbeitserstellung) auf **http://webservices.mii.at/miiCurrencyConverterService/miiCurrencyConverterService.asmx** zu erreichen. Die daraufhin präsentierte WebService-Testseite sieht etwa folgendermaßen aus:

http://webservices.mii.at/miiCurrencyConverterService/miiCurrencyConverterService.asmx :

Service1

The following operations are supported. For a formal definition, please review the Service Description

- convertWithDate
  Converts 'amount' units from 'curr_from' to 'curr_to' using the exchange rates of 'curr_date', supplied in the date format based on 'localeID'. Examples of localeIDs: For the United States: 1033, for Austria: 3079. A list of all possible localeIDs is available at http://support.microsoft.com/support/kb/articles/Q221/4/35.ASP. Note that the conversion is always based on the Austrian exchange rates which are only valid for Austria !
- convert
  Converts 'amount' units of 'curr_from' to 'curr_to', using the latest exchange rates available. Note that the conversion is always based on the Austrian exchange rates which are only valid for Austria !

- getDateOfLatestExchangeRates
  Gets the date of the latest available exchange rates in a date format based on the 'localeID'. Examples of localeIDs: For the United States: 1033, for Austria: 3079. A list of all possible localeIDs is available at http://support.microsoft.com/support/kb/articles/Q221/4/35.ASP.
- getCurrencies
  Returns all currently supported currencies in an ordered array of strings
- getCurrencyFromLCID
  Returns the national currency , if available, corresponding to the supplied localeID (as integer value). Examples of localeIDs: For the United States: 1033, for Austria: 3079. A list of all possible localeIDs is available at http://support.microsoft.com/support/kb/articles/Q221/4/35.ASP.

Der erste Link oben ("**Service Description")** führt auf die formale Beschreibung des Web Services in WSDL. Sie ist in XML formuliert (und daher plattformunabhängig) und beschreibt das Web Service mit seinen Methoden (Aufrufsyntax, Übergabetypen, Rückgabetypen) für jede Benützungsart (HTTP-POST, HTTP-GET und HTTP-SOAP). Aus der WSDL können Entwicklungswerkzeuge eine Proxy-Klasse generieren, über die das Web Service wie ein lokales Objekt angesprochen werden kann, ähnlich Java-RMI, aber auf Basis von HTTP. Eine detaillierte Beschreibung von WSDL ist in [W3C 2001] zu finden.

Aus der WSDL von MCCS geht hervor, dass MCCS folgende Methoden (hier in Java oder C#-Syntax) zur Verfügung stellt (nochmals mit deutschen Erklärungen):

- *public decimal convertWithDate(string curr_from, string curr_to, decimal amount, string curr_date, int localeID)*

  Diese Methode konvertiert 'amount' Einheiten der Währung 'curr_from' in die Währung 'curr_to' zu dem Datum 'curr_date' auf Basis des Datumsformats, das durch localeID spezifiziert ist. Die localeID von Österreich ist z.B. 3079. Eine Liste aller localeIDs ist auf http://support.microsoft.com/support/kb/articles/Q221/4/35.ASP zu finden. Die Kurse basieren immer auf den Kursen der ÖNB und sind daher nur für Österreich gültig !

- *public decimal convert(string curr_from, string curr_to, decimal amount)*

  Diese Methode konvertiert 'amount' Einheiten der Währung 'curr_from' in die Währung 'curr_to' zu zuletzt verfügbaren Kursen. Für die localeID und Gültigkeit der Kurse gilt das unter convertWithDate gesagte.

- *public string getDateOfLatestExchangeRates(int localeID)*

  Liefert das Datum der zuletzt verfügbaren Kurse. Für die localeID gilt das unter convertWithDate gesagte.

- *public string[] getCurrencies()*

  Liefert alle unterstützten Währungen im ISO-3-Zeichen-Code in Form eines geordneten String-Feldes.

- *public string getCurrencyFromLCID(int localeID)*

  Diese Methode liefert die Landeswährung zu einer vorgegeben localeID (sofern verfügbar)

Durch Drücken auf die entsprechenden Links öffnen sich Webseiten, die mittels entsprechenden Formularen die Eingabe der Übergabeparameter ermöglichen. Durch Drücken des „Invoke"-Buttons wird eine Methode ausgeführt. Die Testseite verwendet dazu HTTP-GET. Der Rückgabewert kommt in Form eines XML-Dokumentes über HTTP zurück.

4.2. Einsatz von MCCS

Ein .NET Web Service, wie MCCS, lässt sich über drei verschiedene Arten ansprechen:

- HTTP-GET (beispielsweise über die Test-Seite)
- HTTP-POST
- HTTP-SOAP (oder kurz: SOAP)

Diese Möglichkeiten sollen nun anhand des Beispielaufrufes **convert("DEM","ATS",100)** im Detail erläutert werden.

4.2.1. Aufruf über HTTP-GET und HTTP-POST

HTTP-GET und HTTP-POST erlauben als Übergabeparameter nur primitive .NET Datentypen. Es sind, im Unterschied zu SOAP, keine Klassen, Structs und DataSets möglich.
Als Rückgabeparameter sind hingegen, dank dem XML-Format, alle .NET Datentypen möglich.

Bei HTTP-GET stehen Methodenname und Parameter in der URL:

- Anforderung via HTTP-GET

```
GET
/miiCurrencyConverterService/miiCurrencyConverterService.asmx/conve
rt?curr_from=DEM&curr_to=ATS&amount=100 HTTP/1.1
Host: webservices.mii.at
```

Deutlich kann man die Codierung in der URL mittels '?' und '&'-Symbole erkennen.

- Anforderung via HTTP-POST

```
POST
/miiCurrencyConverterService/miiCurrencyConverterService.asmx/conve
rt HTTP/1.1
Host: webservices.mii.at
Content-Type: application/x-www-form-urlencoded
Content-Length: length (entsprechende Länge in Bytes)
curr_from=DEM&curr_to=ATS&amount=100
```

Hier wird die Parameterinformation über den Content-Type des HTTP-POST Protokolles übergeben.

Die Antwort ist bei HTTP-GET und HTTP-POST gleich:

- Antwort:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length (Entsprechende Länge in Bytes)
<?xml version="1.0" encoding="utf-8"?>
<decimal
xmlns="http://webservices.mii.at/miiCurrencyConverterService/">703.
55</decimal>
```

Das eigentliche XML-Dokument beginnt nach dem HTTP-Header ("`<?xml version ...`").

4.2.2. Aufruf über SOAP

SOAP stellt "einen einfachen Mechanismus zum Austausch von strukturierter und typisierter Information zwischen kommunizierenden Rechnern in einer dezentralisierten, verteilten Umgebung zur Verfügung [Brandstätter 2001]. SOAP basiert auf XML und verwendet HTTP und SMTP als Transportprotokoll.

Der Einsatz für RPC (Remote Provedure Calling) ist nur eine Möglichkeit von vielen: Mittels SOAP sind alle .NET Datentypen als Übergabeparameter möglich, zuzüglich Klassen, Structs und DataSets. Die Anforderung wird mittels HTTP-POST abgewickelt. Im HTTP-Body steht die SOAP-Nachricht:

- SOAP-Anforderung

```
POST /miiCurrencyConverterService/miiCurrencyConverterService.asmx
HTTP/1.1
Host: webservices.mii.at
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"http://webservices.mii.at/miiCurrencyConverterService/convert"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <convert
xmlns="http://webservices.mii.at/miiCurrencyConverterService/">
      <curr_from>DEM</curr_from>
      <curr_to>ATS</curr_to>
      <amount>100</amount>
    </convert>
  </soap:Body>
</soap:Envelope>
```

Das Schlüsselwort `SOAPAction:` definiert das Ziel der SOAP-Nachricht. Die SOAP-Spezifikation [W3C 2000a] definiert das **Envelope**-Element, als Kind-Elemente einen optionalen **Header** und das erforderliche **Body**-Element. Das Envelope-Element umfasst die gesamte SOAP-Nachricht und definiert, was in der SOAP Nachricht enthalten ist, wer sie empfangen soll und welche Teile der Nachricht optional und welche erforderlich sind. Im Header kann die Funktionalität einer SOAP-Nachricht erweitert werden. Typische Beispiele

sind Authentifizierung, Transaktionsmanagement und Zahlungsfunktionen. Das Header-Element wird von MCCS nicht benutzt. Im Body-Element steht die eigentliche Information für den Endempfänger, in diesem Fall RPC-Information zum Aufruf der Methode "convert".

- SOAP-Antwort

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <convertResponse
xmlns="http://webservices.mii.at/miiCurrencyConverterService/">
      <convertResult>703.55</convertResult>
    </convertResponse>
  </soap:Body>
</soap:Envelope>
```

Im Fall von MCCS wäre die Verwendung von SOAP optional, da die Methoden nur einfache Datentypen verwenden.

Für komplexere Anwendungen bietet SOAP aber noch mehr: Der Zugriff auf Web Services kann alternativ auch persistent sein, d.h. Instanzvariablen bleiben erhalten, und außerdem ist ein asynchroner Aufruf von Methoden mit Rückgabewert möglich. Das heißt, dass das aufrufende Programm und eine Web Service- Methode parallel ausgeführt werden können. Der Rückgabewert wird zu einem beliebigen Zeitpunkt über einen Event-Handler bereitgestellt.

Die SOAP-Spezifikation ist zur Zeit noch in Entwicklung, eine Standardisierung ist aber abzusehen [W3C 2000a].

4.3. Programmierung von MCCS

Anhand von MCCS soll kurz die Programmierung eines Web Services beschrieben werden.

Es gibt für .NET bereits sehr viele Sprachendialekte, die Umstellung vorhandener Objektimplementierungen auf Web Services gestaltet sich jedoch bei allen sehr einfach.

MCCS wurde in C# implementiert, einer Sprache, die syntaktisch Java sehr ähnlich ist und von Microsoft als die wichtigste .NET-Programmiersprache propagiert wird.

Klassen und Methoden können in dieser Sprache sogenannte Attribute zugewiesen werden. Sie sind unmittelbar vor der Klasse oder vor den Methoden in eckigen Klammern ([*Attributname*(*Option=Wert*, *Option=Wert*,...)]) anzugeben.

Die Schritte zur Erzeugung eines Web Services sind folgendermaßen:

- Anlegen einer .asmx Datei (enthält Web Service-Direktive und Initialisierungscode und dient als Template)
- Das Hinzufügen des Attributs "WebMethod" zu einer Public-Methode macht sie automatisch zu einer Web Methode
- Optional: "WebService"-Attribut für die Klasse, Namespace- und SoapAction-Attribut für SOAP

ASP.NET erledigt den "Rest": Erzeugen der Service-Beschreibung als WSDL-Datei und Bereitstellung der WebService-Testseite (mittels compile on demand).

Der folgende Sourcecode-Auszug aus miiCurrencyConverterService zeigt keine Implementierungsdetails der MCCS-Funktionalität, sondern soll vor allem die Unterschiede zu einer gewöhnlichen Objektimplementierung aufzeigen (Unterschiede sind fett gedruckt):

```
   //
   // .NET WebServices am Beispiel von 'miiCurrencyConverterService'
   //

   namespace miiCurrencyConverterService
   {
      // Für WebServices benötigte Klassen:
      ...
      using System.Web;
      using System.Web.Services;
      ...
      //

      [WebService(Namespace="urn:schemas-mii-
at/miiCurrencyConverterService",Description="This WebService exposes
WebMethods for converting monetary units based on daily actual exchange rates.
")]
      public class miiCurrencyConverterService :
System.Web.Services.WebService
      {

            // Konstantendeklaration
            ...
            //

            // Vom ASP+ WebService Designer genriert

            public miiCurrencyConverterService()
            {
                  //CODEGEN: This call is required by the ASP+ Web Services
Designer
                  InitializeComponent();
            }

            /// <summary>
            ///    Required method for Designer support - do not modify
            ///    the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
            }

            /// <summary>
            ///    Clean up any resources being used.
            /// </summary>
            public override void Dispose()
            {
            }

            // Auszug : 'convert' - Methode
            //
            // public decimal convert (string curr_from,string curr_to,
decimal amount)
            //
```

```
            [SoapMethod(Action="urn:schemas-mii-
at/miiCurrencyConverterService#convert",
            RequestNamespace="urn:schemas-mii-
at/miiCurrencyConverterService",
            ResponseNamespace="urn:schemas-mii-
at/miiCurrencyConverterService")]
            [WebMethod(Description="Converts 'amount' units of 'curr_from' to
'curr_to', ...")]
            public decimal convert (string curr_from,string curr_to, decimal
amount)
            {
                  decimal c1 , c2 ;

                  // Fehlerbehandlung für Beträge < 0
                  if (amount < 0)
                  {
                        // Exception über SOAP !
                        throw new
CurrencyConvertArgumentOutOfRangeException(ERROR_MSG1);
                  }

                  // Hashtabelle holen:
                  // Schlüssel = Währungskürzel
                  // Wert = Umrechnungsfaktor zu Euro

                  Hashtable tempHash =
getCurr(DateTime.Parse(getDateOfLatestExchangeRates(0),
DateTimeFormatInfo.CurrentInfo));

                  // Währung unbekannt: Fehlerbehandlung ähnlich wie oben
                  ...
                  //

                  // Euro - Umrechnungsfaktoren holen

                  c1 = (decimal) tempHash[curr_from];
                  c2 = (decimal) tempHash[curr_to];

                  ...

                  // Umrechnen und Ergebnis zurückliefern

                  return Decimal.Round((amount/c1)*c2,2);

            } // Von 'convert' Methodendeklaration

      // Restliche Methoden (getCurrencies, ...)
       ...
      //

      } // Von Class - Deklaration

   } // Von Namespace-Deklaration
```

Tatsächlich lassen sich auf diese Art herkömmliche Objektimplementierungen (mit Business-Logik) sehr einfach auf Web Services umstellen.

4.4. Zusammenfassung MCCS

Bei MCCS handelt es sich um "ASP-Lite":
Es ist ein kostenloser ASP Dienst, daher ohne Einsatz von Verrechnungstechnologie. So sind in der derzeitigen Version auch keine SLAs vorgesehen. MCCS soll als Beispiel für einen einfachen, sinnvollen Einsatz von internetbasierten Applikationsdiensten fungieren.

Nun zu den technischen Aspekten: MCCS besitzt eine Anbindung an das **Serverdateisystem** (zum Abruf der Kurse), die allerdings für den Benutzer verborgen bleibt (eine sog. "Back-End"-Anbindung). Die Anbindung an eine **Datenbank** ist nicht vorhanden.

Der **Interaktionslevel** ist sehr niedrig, das **Benutzerinterface** (Test-Seite) sehr einfach.

Die **Internetfähigkeit** ist sehr ausgeprägt: MCCS lässt sich über HTTP-GET, HTTP-POST, und HTTP-SOAP nutzen und jedes dieser Protokolle lässt sich auch verschlüsselt über SSL (Secure Socket Layer) "tunneln".

Die **Geräteunabhängigkeit** ist sehr gut: Auf Clientseite lässt sich das Web Service von allen Systemen integrieren, die eine HTTP Verbindung unterstützen. Die Test-Seite nimmt Rücksicht auf die Fähigkeiten des vorhandenen Browsers. Besonders einfach gestaltet sich die Benutzung bei Verwendung von XML- oder SOAP-Toolkits. Auch serverseitig lässt sich das Web Service sehr leicht auf andere Systeme umstellen, sobald .NET auch für andere Plattformen verfügbar ist. So sind auch die **Integrationsmöglichkeiten** vielfältig: Das Web Service kann in Software integriert werden, die für eigene Zwecke bestimmt ist, oder als ASP-Dienst angeboten wird.

Da der Zugriff auf das Web Service anonym ist, gibt es keine **Personalisierungsmöglichkeiten**. Da kein Sessionzustand gespeichert werden muss, ist auch keine **Sessionverwaltung** erforderlich.

Zur Zeit werden im Rahmen der "Global XML Architecture" (GXA) **Kommunikations-Mechanismen** zur Verbesserung der Sicherheit von SOAP Nachrichten entwickelt [Microsoft 2002e]. Sobald die darin enthaltene "Web Service Security Language" (WS-Security) standardisiert ist, ließe sich das Web Service hinsichtlich Kommunikationssicherheit mittels digitaler Signaturen und Verschlüsselung erweitern.

Die **Applikationsprogrammlogik** ist sehr einfach gehalten: Sie besteht aus vier Dateien, eine Datei (miiCurrencyConverterService.asmx) enthält die Web Service-Objektimplementierung. Sie besteht aus einer Sammlung von Methoden und Web Service-Methoden, aber es ist kein Hauptprogramm vorhanden. Die anderen drei Dateien enthalten die Implementierungen von speziellen SOAP-Exceptions.

Über **Verfügbarkeit** und **Skalierbarkeit** sind nur Erfahrungswerte vorhanden. Softwaretechnisch sind jedenfalls keine Maßnahmen vorhanden, um **Hochverfügbarkeit** zu garantieren (d.h. keine Verwendung des Application Center Servers). Die Skalierbarkeit wird jedenfalls verbessert, wenn MCCS der einzige Internetdienst auf dem Server ist, denn dann kann der Server nicht durch andere Aufgaben blockiert sein und die Datei mit den Kursen ist immer im Cache. Die **Antwortzeit** wird dadurch auch vorhersehbar und die **Performance** ist leicht zu testen. Über eine Modemverbindung innerhalb Wiens geschieht eine Währungsumrechnung in weniger als 0.5 Sekunden. Die **Verfügbarkeit** ist nach eigenen Tests gut, denn das Web Service war und ist bis zum heutigen Zeitpunkt (30.1.2002) Tag und Nacht erreichbar. Allerdings ist die Test-Seite derzeit nicht aktiv.

Außer der Test-Seite gibt es keine **Benutzerinterfaceprogrammlogik**, da es die ausschließliche Funktion eines Web Services ist, Business Logik zur Verfügung zu stellen. Die Benutzerinterfaceprogrammlogik muss je nach Anwendung erst implementiert werden. Dies macht den Dienst aber sehr flexibel.

Die **Sicherheit** des Webzugriffes wird nur durch Verwendung von SSL oder der digitalen Signierung der SOAP-Nachrichten gewährleistet. Es ist aber keine Möglichkeit bekannt, Serverressourcen oder Daten durch MCCS zu missbrauchen. Lediglich eine "Denial of Service"-Attacke (Überlastung des Servers) wäre denkbar, da am Server aber keine anderen Dienste laufen und keine Garantie über den Zugriff auf MCCS gewährleistet wird, ist ein solcher Missbrauch nicht bedrohlich. Durch das Fehlen einer Benutzerinterfaceprogrammlogik ist auch eine Gefährdung der Sicherheit des Endgerätes auszuschließen. Der Benutzer hat immer volle Kontrolle, welche Daten er an MCCS übergibt.

Die **Benutzerfreundlichkeit** ist bei Verwenden der Test-Seite befriedigend. Noch komfortabler wäre das Web Service aber über einen speziellen Client zu benutzen. Clients können z.B. mittels "MS Soap Toolkit" auf einfache Weise erstellt werden. Mittels dieses Werkzeugs ist es möglich, MCCS wie ein lokales Objekt zu verwenden.

Kapitel 5

Zusammenfassung und Ausblick

Application Service Providing (**ASP**) bedeutet die Vermietung und Bereitstellung von Anwendungsfunktionalität von einem Rechenzentrum über Internet oder ein privates Netzwerk für mehrere Kunden. Die **Merkmale** von ASP sind eine zentrale Speicherung, ein webbasierter Zugriff für den Anwender, eine minimale Anpassung der Applikationen an einzelne Kunden, ein Mietvertrag, der vertraglich zugesicherte Leistungen (Service Level Agreements) enthält und die Bezahlung nach Zeit und/oder Inanspruchnahme.

ASP hat sich aus den neuen wirtschaftlichen Bedürfnissen und technischen Möglichkeiten von IT-Outsourcing, Application Hosting und Portal Computing entwickelt. Dabei nahm IT-Outsourcing Einfluss auf das (Geschäfts-) Modell von ASP, Application Hosting lieferte die Grundlage für die Verbreitung von ASP und Portal Computing lieferte das Fundament für die technische Grundlage von ASP.

Die **Vorteile** aus **Anwendersicht** gegenüber herkömmlicher Softwarenutzung sind Zeit-, Kosten- und Personalreduktion, variable Kosten statt IT-Investitionen, bessere Produktvergleichbarkeit, Geräte- und Ortsunabhängigkeit, die hohe Technologiekompetenz des Dienstleisters und die aus all eben genannten Vorteilen resultierenden besseren Marktchancen für unternehmerisch tätige ASP-Kunden.

Dem gegenüber stehen **Nachteile**: wenig Marktreife, nicht eindeutige Kostenreduktionen, starke Abhängigkeit von Application Service Provider (ASP) und Internet, hohe Kundenbindung, Datenschutz- und Datensicherheitsprobleme, kaum standardisierte Schnittstellen und die Beschränkung auf die Fähigkeiten der Internettechnologien (Browser, Protokolle etc.).

Chancen, Risiken bzw. Herausforderungen für ASPs sind die Neuausrichtung der Geschäftsmodelle und der starke Wettbewerb, die Partner- und Kundenorientierung, das notwendige Branchenwissen bei branchenspezifischen Applikationen, die Regelung von Urheberrechten und Markenzugehörigkeiten, die hohe Verantwortung gegenüber den Kunden, das Vertrauen zu ASP-Partnern und deren Kontrolle und die Bewältigung von Sicherheitsproblemen wie Softwarepiraterie und Hacker.

Am ASP-Geschäft sind in der Regel mehrere **Anbieter** wie Netzwerkanbieter, Application Infrastructure Provider (AIPs), Hoster, Soft- und Hardwarehersteller, Independent Software Vendors (ISVs), Service Integratoren sowie Beratungsfirmen beteiligt. Der ASP schließlich führt die Leistung dieser Anbieter zusammen und betreut die Beziehung zu den Endkunden. ASPs selbst lassen sich nach horizontaler oder vertikaler Ausrichtung, nach der Art oder der Komplexität der angebotenen Applikationen und Dienstleistungen oder nach der Art ihrer Geschäftsmodelle beschreiben.

Typische ASP-Applikationen sind analytische Anwendungen, vertikale Applikationen, ERP-Software, CRM-Software, kollaborative Anwendungen sowie Personalapplikationen. Die Zielgruppe von ASP stellen in erster Linie die kleinen und mittleren Unternehmen (KMUs) dar, denen eigene IT-Abteilungen oder IT-Spezialisten fehlen bzw. die sich durch das ASP Modell teure Software ohne hohe Initialinvestitionen leisten können.

Die aktuelle Marktsituation und die **Marktentwicklung** von ASP ist von nahezu ständigem Wachstum geprägt, während der europäische Markt dem Markt in den USA um etwa zwei Jahre hinterherhinkt. Nachdem die erste Phase der Marktentstehung in den Jahren 2000, 2001 und 2002 vorübergegangen ist, werden sich die ASPs in den Jahren 2002 und 2003 durch starke Kompetenzen und hohe Umsätze auszeichnen. In dieser Zeit werden jedoch auch etwa 60% aller derzeitigen ASPs vom Markt verschwinden. Ab 2004 werden sich etablierte Geschäftsmodelle und ERP- bzw. CRM-Angebote durchsetzen.

Die **betriebswirtschaftlichen Gründe**, einen ASP einzusetzen, sind niedrige Initialinvestitionen, Minimierung der Total Cost of Ownership (TCO), gute Kostentransparenz, Verbesserung der Personaleffizienz, geringe Kapitalbindung und langsamerer Anstieg der Gesamtkosten.

Die Abhandlung **rechtlicher Aspekte** steht noch am Anfang: ASP ist jedenfalls Miete im Sinne von Gebrauchsüberlassung von Software. Als Nebenleistungspflichten können in den Service Level Agreements (**SLAs**) Instandhaltung, Instandsetzung, Beseitigung von Softwarefehlfunktionen, Updates, Systembackups, Verfügbarkeitszusicherungen sowie ein Kundendienst definiert sein. Zu den Pflichten der Gewährleistung und Haftung zählen die mängelfreie Überlassung der Software, die Erhaltung der Gebrauchstauglichkeit und eine Beratungs-, Sorgfalts- und Schutzpflicht. Bei Nichteinhaltung hat der Kunde Anspruch auf Mängelbeseitigung bzw. Schadensersatzanspruch. Pflichten des Kunden sind Zahlungspflicht, die Rückgabe der Software nach Beendigung der Mietzeit, die Vermeidung von Schäden an der Software sowie der Schutz der Programme vor unberechtigten Zugriffen Dritter. Damit der ASP urheberrechtlich geschützte Software einsetzen darf, ist die Einräumung von Nutzungsrechten (Lizenzen) vorgesehen. Der ASP muss sich das Recht der öffentlichen Wiedergabe sowie das Vervielfältigungsrecht einräumen lassen. Der ASP muss weiters die Datenschutzbestimmungen befolgen.

Die Beschreibung des ASP-Modells im ersten Kapitel liefert unter anderem auch eine Entscheidungsgrundlage für oder gegen das ASP Modell für Anbieter und Softwareanwender.

Im Hauptteil dieser Diplomarbeit wurden die organisatorischen und technischen Konzepte, Voraussetzungen und Herausforderungen für die Anwendungsbereitstellung über Internet aufgezeigt. Die zentrale Softwarekomponente für die internetbasierte Anwendungsbereitstellung ist die **ASP Basistechnologie**. Als konzeptuell notwendige Grundvoraussetzungen für eine Bereitstellung von Software über Internet wurden **Internetfähigkeit und Kommunikation**, **Trennung von Benutzerinterfacelogik und Applikationslogik**, **Mehrbenutzerfähigkeit und Zugriffsverwaltung**, **Plattformunabhängigkeit und Integration** sowie **Verrechnungstechnologien** herausgearbeitet. Eben aufgezählte Funktionalitäten stellen insoferne eine Besonderheit dar, als dass sie bei herkömmlicher lokal installierter Software nicht zwingend sind.

Zur Bewertung des ASP Betriebes gibt es weiters Kriterien wie **Server Architektur**, **Applikationsprogrammlogik**, **Verfügbarkeit**, **Performance**, **Skalierbarkeit**, **Sicherheit** und **Operation Management**. Ebenfalls wurde die Schnittstelle zur Anwendung, der Client, hinsichtlich **Benutzerinterface-Programmlogik**, **Benutzerfreundlichkeit und Anpassbarkeit**, **Mächtigkeit** sowie **Sicherheit** untersucht. Eben genannte Kriterien wurden in der Diplomarbeit mit **Server Herausforderungen** und **Client Herausforderungen** übertitelt, da diese genau die technischen Herausforderungen für einen erfolgreichen ASP Einsatz darstellen. Die Qualität eben dieser Leistungen drückt sich in der Höhe der **Service Levels** aus. Hohe Service Levels bilden eine Schlüsselrolle für qualitative ASP Angebote.

In Theorie und Praxis gibt es zwei sehr unterschiedliche ASP Basistechnologien: **websprachenbasierte** und **terminalbasierte Systeme**. Der detaillierte Vergleich dieser

Technologien und die Ausarbeitung der **Eignung für ASP in Theorie und Praxis** war eine der Hauptaufgaben dieser Diplomarbeit.

Systeme auf Websprachenbasis bestehen typischerweise serverseitig meist aus einem Web Server oder einem Application Server, aus in speziellen Websprachen implementierten Softwarekomponenten und aus einem Datenbankserver. Auf Clientseite wird nur ein Webbrowser vorausgesetzt. Applikationen auf Websprachenbasis senden Benutzerinterfacelogik, die in prozeduralen (clientseitige Websprachen wie JavaScript) oder deklarativen Websprachen (Beschreibungssprachen wie HTML) implementiert sein kann, zum Client. Die Trennung von Benutzerinterfacelogik und Applikationslogik geschieht zur Design- und Implementierphase. Die Kommunikation zwischen Server und Client wird über HTTP oder HTTP-SOAP abgewickelt.

Die Kennzeichen von Applikationen auf Basis von Websprachen sind eine aufwendige, spezielle Implementierung von Applikationsprogrammlogik und Benutzerinterfaceprogrammlogik für den browserbasierten Betrieb, hoher Aufwand für Sessionverwaltung durch die Beschränkung auf das sessionlose HTTP-Protokoll, niedrige Anforderungen an Netz- und Serverressourcen, niedrige Kosten für Verfügbarkeit und Skalierbarkeit durch einfache Möglichkeiten der Lastverteilung, mittelmäßige Plattformunabhängigkeit aufgrund zueinander inkompatibler Browser, sehr gute Unterstützung von datenbankorientierten Applikationen, unzureichende Unterstützung von dateiorientierten Applikationen, einfache, wenig interaktive Benutzerschnittstellen aufgrund der Seitenorientierung von HTML sowie Möglichkeiten zur automatischen Anpassung an die Fähigkeiten des Endgeräts und damit gute Unterstützung von mobilen Geräten. Durch Verwendung von Applikationsservern statt Web Servern kann eine bessere Wartbarkeit, Skalierbarkeit, Verfügbarkeit und Sicherheit sowie ein besseres Server Management erreicht werden.

Systeme auf Terminal-Basis bestehen vereinfacht betrachtet aus dem Terminalserver, beliebigen auf dem jeweiligen Server ausführbaren Applikationen sowie aus Dateiservern und Datenbanken. Auf Clientseite sind nur spezielle Terminalclients, die eigenständig am Endgerät oder in dessen Browser laufen, notwendig. Applikationen auf Terminalbasis laufen komplett am Server und übertragen nur die Bildschirmausgabe, Tastaturanschläge und Mauseingaben über das Netz. Die Trennung von Benutzerschnittstelle und Applikationsprogrammlogik geschieht also zur Laufzeit. Die Kommunikation zwischen Client und Server wird über Terminalprotokolle wie ICA, RDP oder AIP abgewickelt.

Applikationen, die über Terminalbasistechnologien bereitgestellt werden, können beliebige Windows- oder UNIX-Anwendungen sein, bieten ausgezeichnetes, 100%ig serverseitiges Sessionmanagement, stellen relativ hohe Anforderungen an Netz- und Serverressourcen, erfordern hohe Kosten für Verfügbarkeit und Skalierbarkeit aufgrund wenig Möglichkeiten zur Lastverteilung, bieten je nach Hersteller mehr oder weniger gute Plattformunabhängigkeit, eine sehr gute Unterstützung von datenbank- und dateiorientierten Applikationen, komplexe, hochinteraktive Benutzerschnittstellen wie bei einer lokalen Applikation, wenig Möglichkeiten zur Anpassung an die Fähigkeiten unterschiedlicher Endgeräte und daher eine schlechte Eignung für mobile Geräte. Applikationen auf Basis von Terminaltechnologien sind komfortabler zu bedienen, sind jedoch für den ASP teurer zu warten.

Welcher Technologie der Vorzug gegeben werden soll, hängt nahezu ausschließlich von den Anforderungen der zu betreibenden Applikation und deren Benutzer ab. Es scheint jedoch so zu sein, dass insbesondere E-Business- und ERP-Software verstärkt als websprachenbasierte ASP-Lösung angeboten werden, da diese Applikationen schon vor der Einführung des ASP-Modells webbasiert zur Verfügung standen. Hoch interaktive Software wie Office, CAD, etc. sind jedoch ausschließlich für die Verwendung per Terminal-Basistechnologien geeignet.

Die **Eignung von ASP Basistechnologien in der Praxis** wurde im **praktischen Teil** untersucht.

Citrix MetaFrame XPe bietet derzeit die beste technische Grundlage für ASP, dies sowohl im Hinblick auf die Anforderungen von Applikationen als auch im Hinblick auf hohe Service Levels. Nur bei Server Integration, Skalierbarkeit, Server-Wartbarkeit und der Unterstützung von mobilen Geräten wie z.B. Handys ist Citrix den **Application Servern** unterlegen. Grenzbereiche sind Hochverfügbarkeit, Mehrbenutzerfähigkeit und Performance. Diese Qualitäten können mit Citrix MetaFrame XPe nur mit etwas höherem Aufwand und höheren Kosten bewerkstelligt werden. Vergleichsweise schlecht schneiden **traditionelle Internetsprachen** gemeinsam mit herkömmlichen Web Servern ab.

Als **Beispiel für eine praktische ASP-Anwendung** auf Basis von **Microsoft .NET** ist im Rahmen der Diplomarbeit ein Web Service namens **miiCurrencyConverterService** entstanden. Das Web Service stellt tagesaktuelle Kurse zur internetbasierten Währungsumrechnung zur Verfügung. Außer, dass die Anwendung kostenlos ist und dass keine Service Level Agreements existieren, ist sie durchaus als ASP Lösung einzustufen. Anhand dieses Beispiels ist auch sehr gut zu sehen, dass die Anforderungen an ASP Technologien in erster Linie von den Anforderungen der anzubietenden Applikation abhängen, denn der gezeigte Währungsumrechner stellt wesentlich geringere Ansprüche an die ASP Basistechnologie als z.B. eine ERP-Lösung.

In Zukunft werden die ASP Basistechnologien sicherlich an Leistungsfähigkeit zulegen und ASPs werden höhere Service Levels garantieren können. Das Vertrauen in den ASP Betrieb und die Kundenzufriedenheit werden steigen. Einen Schwerpunkt der zukünftigen Entwicklungen setzen die mobilen Applikationen. Hier stecken die derzeitigen Angebote noch in den Kinderschuhen.

Als eine für mich persönlich interessante Frage erachte ich, ob trotz der Vormachtstellung von Websprachen und Terminaltechnologien eine Symbiose aus ASP Basistechnologien zu erwarten ist.

Die rasante Marktentwicklung von ASP lässt jedenfalls auch auf eine intensive technologische Weiterentwicklung hoffen.

Anhang

Interview über technische Aspekte von ASP

Das vorliegende Interview wurde am 8.12.2001 mit Herrn Dominik PAIHA, zum Zeitpunkt des Interviews als Senior Cosultant bei MS Österreich beschäftigt, geführt. Dabei lag der vorliegende schriftliche Fragenkatalog zugrunde. Da zum damaligen Zeitpunkt kein Diktiergerät zur Verfügung stand, sind die Aussagen nur in Stichwörtern niedergeschrieben worden.

Daher sind die Antworten sinngemäß und keinesfalls wortwörtlich zu verstehen.

Allgemeines über ASP Basistechnologien

Kann man behaupten, dass es prinzipiell zwei unterschiedliche Technologien, nämlich terminalbasierte- und websprachenbasierte Systeme gibt ? Gibt es noch andere Konzepte ?

Ja, das kann man so sagen. Auch WAP kann man zu den Websprachen rechnen.

Terminals

Ist ASP über ein Terminal immer sessionbasiert ? (d.h. eine gesamte Applikationssitzung vom Start bis zum Beenden einer Applikation ist EINE Session) ?

Richtig, bei WTS und Citrix läuft am Client ein ActiveX-Script oder ein Netscape-Plug-In, das statt dem verbindungslosen HTTP-Protokoll ein eigenes Protokoll benutzt (ICA oder RDP). Sowohl WTS, Citrix als auch Tarantella können auch pushen, d.h. eine Nachricht an den Client senden, ohne dass eine explizite Anforderung vorausging.

Am Client wird kein Sessionzustand gespeichert. Wird der Client beendet, oder stürzt er ab, dann kann er sich innerhalb von zehn Minuten wieder verbinden. Die Erkennung der Applikationssession erfolgt durch den Username. Nach diesem Timeout wird die Session beendet.

Wir haben bei einem Kunden ein Feature implementiert, welches bei allen Office-Applikationen nach diesen zehn Minuten automatisch alle offenen Dokumente speichert. Seit Office XP ist das nun sauber möglich. Damit brauchen die Anwender keinen Datenverlust mehr zu befürchten.

Welches sind derzeit die wichtigsten oder erfolgreichsten terminalbasierten Systeme für ASP ? (WTS, Citrix, Tarantella, etc. ?)

Citrix und Microsoft (WTS) haben eine enge Partnerschaft. Die Zukunft wird zeigen, ob diese Produkte zusammenwachsen werden. Tarantella ist meiner Meinung nach nur für

Spezialanwendungen interessant. WTS wird bei Windows NT4 und Windows 2000 mitgeliefert, deshalb wird es recht häufig eingesetzt. Für Citrix ist eine WTS Lizenz Voraussetzung, deshalb muss für ein anderes Betriebsystem zuerst eine WTS-Lizenz erworben werden.

Stimmt es, dass etablierte Software (d.h. Software, die schon lang am Markt ist) fast ausschließlich über terminalbasierte Systeme angeboten wird und nicht auf websprachenbasierten Systeme neu implementiert wird ?

Ja, es ist die schnellste Möglichkeit für den ASP, Software anzubieten. Es öffnet den Markt für große Softwareapplikationen. Webbasierte Systeme sind auch problematisch, da man mit der Sicherheit an HTTP gebunden ist.

Nur Spezialsoftware wie z.B. eine Ticketverwaltung ist für den webbasierten Betrieb geeigneter. Dahinter stehen große Datenbankserver. Mit Terminalservern wäre so ein System zu teuer zu warten.

Was sind die gravierendsten Vorteile und Einschränkungen von terminalbasierten ASP ?

ICA wird von einer breiten Plattform unterstützt. Bei den Websprachen hingegen liegt es in der Hand des Softwareherstellers der jeweiligen Applikation, welche Browser er unterstützt.

Die Einschränkungen sind, dass eine clientseitige Software installiert werden muss, dazu benötigt man einmalig Administratorrechte. Terminals erfordern eine bessere Bandbreite. Der Vorteil von webbasiertem ASP ist, dass der Client keine Wartung braucht.

Websprachen

Was sind die wichtigsten Konzepte (CGI, Enterprise Java Beans etc.), Programmier- bzw. Websprachen (Java Servlets, Java Applets, JavaScript, PHP etc.) und Plattformen (.NET, BEA WebLogic etc.) ?

CGI ist ein Auslaufmodell, da die Security unzureichend ist.

Java Beans ist von der Idee her gut, die Plattformunabhängigkeit ist jedoch serverseitig nicht gegeben (Unterschiede z.B. im Dateisystem von UNIX und Windows).

Der Nachteil von EJB ist die schwache Performance.

Bei Websprachen wären als wichtige zu nennen: Active Server Pages, VBScript, Jscript, JSP (Servlets), PHP und Perl, wobei letzteres nicht zu unterschätzen ist !

Wichtige Plattformen sind IBM WebSphere, Microsoft .NET, Iplanet und BEA WebLogic. BEA ist von allen EJB Servern der performanteste.

Ist die Laufzeitumgebung fast immer der Browser über HTTP ?

Ja. Die Ausnahme betrifft WAP, hier kommt, wenn man so möchte, der "Browser" des Handies zum Zug.

Ist der Hauptunterschied zum terminalbasierten System, dass Semantik übertragen werden kann ?

Ja.

Kann man der Seitenbasiertheit von Browsern bzw. der Sessionlosigkeit von HTTP noch anders als durch ein JAVA-Applet entkommen ? (Sessions z.B. durch URL codiert oder mit Cookies - Probleme dabei ?)

Der Seitenorientierung kann man nicht entkommen.

Fortschrittliche Plattformen wie MS .NET und BEA Weblogic trennen nicht mehr strikt zwischen Server- und Clientcode, sondern zwischen Benutzerinterface- und Applikationsprogrammlogik, der Client-Code für die Kommunikation wird dabei automatisch erzeugt, ebenso wie Code zum Verwalten von Sessioninformation. Wie begegnet man aber dem Problem der Seitenbasiertheit ?

Entweder die Programmierer der Plattformen oder die Programmierer der Webapplikationen müssen sich mit diesem Problem auseinandersetzen.

Hat das HTTP Protokoll, und wie es ein Browser z.Z. interpretiert, noch Zukunft für ASP ?

Ja, der Hauptteil bei HTTP ist: ICH will INFORMATIONEN. Unter diesem Gesichtspunkt gibt es eine Reihe von Applikationen.

Citrix ist eigentlich nur eine Verlagerung von Wartung. Dieser Wartungsaufwand ist bei Citrix höher. Deshalb wird es HTTP nicht so leicht verdrängen.

Warum wird bei Websprachen aus Sicht des Clients fast immer auf eine Anbindung zum Serverdateisystem verzichtet, wo doch bei Citrix das Serverdateisystem fixer und wesentlicher Bestandteil ist ? Warum werden "als Ersatz" meist Datenbanken verwendet ? Kann man sagen, dass eine Anbindung zu einer Datenbank sowohl client- als auch serverseitig wesentlich performanter ist ?

Man verwendet die Datenbank wegen ihrer Transaktionalität. Das bedeutet, dass man immer einen konsistenten Zustand der Datenbank hat. Man weiß immer: "Es ist gut gegangen oder nicht". Bei Abstürzen des Clients ist eine Datenbank einfacher zu handhaben als ein Dateisystem.

Bei Citrix macht man es so, dass mehrere Rechner zu Cluster zusammengeschaltet werden. Daten werden auf Dateiservern gespeichert, die durch das Clustering ein Rollback beherrschen.

Die Applikationssession läuft jedoch immer nur auf einem dezidierten Server. Der Nachteil ist, dass falls ein Server abstürzt, dann für alle diejenigen Benutzer die Session verloren ist, die auf diesem Server eine offen hatten. Die Anbindung an Dateisysteme ist, kurz gesagt, teurer.

Bei webbasierten Systemen ist das Load-Balancing einfacher. Man bezieht die Sessioninformation über die Datenbank.

Anforderungen an ASP Basistechnologien

Internetfähigkeit und Kommunikation

Bei terminalbasierten Systemen wird der Internetzugriff vollständig von dem der Applikation übergeordneten Terminalprogramm erledigt. Wie funktioniert das Terminalmessaging (z.B. von Citrix) prinzipiell ?

Am Server wird dazu ein virtueller Bildschirmtreiber installiert. Es wird nur Bitmap-Information übertragen. Bei ICA wird die grafische Differenz zum vorhergehenden Frame gesendet. Deshalb dauert es länger, wenn sich die Bildinformation gänzlich ändert. Zusätzlich gibt es im Protokoll sog. "Virtual Channels". Damit lässt sich Zusatzinformation wie Druckerinformnation, QoS-Parameter und Aufforderungen an den Client via Dialogboxen

realisieren. Letzeres ist ein Beispiel für Push/Pull-Verhalten. Es kann Aufforderungen an das Clientbetriebsystem geben.

RDP5.0 ist von der Funktionalität ICA sehr ähnlich. Es ist standardmäßig komprimiert.

Ist es korrekt, dass bei Websprachen die Überlegung, welches Protokoll (verschlüsselt oder nicht, HTTP oder spezielles Protokoll, verbindungsorientiertes oder verbindungsloses Protokoll) verwendet werden soll, schon während der Applikationsdesignphase maßgeblich ist ?

Richtig, bei den Websprachen wird entweder HTTP oder HTTPS eingesetzt. HTTPS wird allerdings heute sinnvoller in Hardware implementiert.

Native Sockets mit eigenen Protokollimplementierungen verwendet man eigentlich nur im Intranet, weil die meisten Firewalls auf Port 80 nur HTTP(S) durchlassen. Bei den Terminals wird zwar auch ein anderer Port verwendet, aber die verwendeten Protokolle können verschlüsselt werden.

Multiuserfähigkeit

Wichtige Punkte beim Ressourcensharing ?

Bei den webbasierten Systemen wird meist eine Datenbank gemeinsam benutzt. Schreibvorgänge bewirken dabei einen Push auf alle Server. Um eine Datenbank zu erweitern kauft man einfach Storageplatz dazu. Da Datenbanken dynamisch skalieren, ist diese Lösung relativ billig.

Terminals erfordern einen höheren Aufwand: Schreibvorgänge auf Dateiserver werden nicht gleichzeitig auf alle Server getätigt. Ausfallssicherheit herzustellen ist sehr teuer und aufwendig. Etwa vier bis fünf Millionen ATS für Hochverfügbarkeit für etwa 100 User in einer Citrix-Umgebung.

Unterschied: Personalisierte Daten (in einer Datenbank oder in einem Serverdateisystem) werden gebraucht oder nicht gebraucht ?

Das System muss multisessionfähig sein. Windows 2000/XP kann das (NT4 benötigt zusätzlich den Terminalserver). Die Applikation muss selbstverständlich ASP-ready sein, sonst muss man dezidierte Server (d.h. ein Server pro Benutzer) einsetzen.

Welche Schwierigkeiten sind bei der Anpassung an ASP mittels terminalbasierte Systeme bei nicht multiuserfähigen Applikationen zu erwarten ? (z.B. personalisierte Programmvoreinstellungen) ?

Nicht-benutzerspezifische Programmvoreinstellungen müssen im Shared-Directory abgebildet werden. Citrix kann nur beim Programmstart Grundvoreinstellungen laden.

Interoperation und Integration

Welche für ASP relevanten Technologien kann/darf man als standardisiert ansehen ? (HTTP, HTML, Java und JavaScript ?) Ist man aber beim Einsatz dieser Technologien ein für alle Mal auf den Einsatz in einem Internetbrowser beschränkt ?

HTTP und HTML sind zwar standardisiert, aber HTML wird von keinem Browser sauber implementiert. Dies ist ein Nachteil der Websprachen. XHTML, die neue XML-konforme

Ausführung von HTML, wird nur von Opera unterstützt. Bei Opera kann ich einstellen, als welcher Browser ich mich ausgeben möchte.

Java ist theoretisch standardisiert, praktisch nein. Das Problem liegt in den Unterschieden der verschiedenen Plattformen (Dateisysteme) und dem nötigen Aufwand in der Implementierung. Anders gesagt: Java bietet zwar alle Vorraussetzungen, die von Programmierern durchgeführten Implementierungen sind allerdings mangelhaft.
Bei JavaScript ist derzeit ECMA 2.6.3 als Standard definiert.

Quasi-Plattformunabhängigkeit: Der Hersteller bietet für sein Produkt so viele Anpassungen für unterschiedliche Systeme, sodass eine nahezu vollständige Plattformunabhängigkeit erreicht wird. Sind die Internetbrowser und Citrix zwei solche gute Beispiele ?

Die Internetbrowser bieten keine gute Plattformunabhängigkeit. So verhalten sich die Internetbrowser auf dem PC anders als auf dem Apple, auch wenn beide dieselbe Versionsnummer tragen.

Bei Citrix ist die Anpassung weitaus besser !

Die grafische Ausgabe bei terminalbasierten Systemen ist sehr starr, für mobile Geräte schlecht geeignet. Ist Plattformunabhängigkeit bei Terminals deshalb weniger gut zu erreichen als durch websprachenbasierte Systeme ?

Ja, man muss von einer Mindestfunktionalität auf Clientseite ausgehen. Ich kann nicht festlegen, wie sich das Benutzerinterface darstellt. Für Pocket PCs ist ein ICA-Client fast nicht sinnvoll !

Aber auch die Websprachen sind hier nicht wesentlich besser: Obwohl ich mit dem Browser verschiedene Einstellungen (z.B. Schriftgröße) ändern kann, kommen viele schlecht gestaltete Anwendungen nicht mit solchen Änderungen klar. Das Layout des GUI kommt durcheinander.

Was ist unter Integration im Zusammenhang mit ASP zu verstehen und mit welchen Aufgabenstellungen ist man in der Praxis konfrontiert ?

Bei den webbasierten Systemen wird dieses Wort nicht benutzt. Bei den Terminals bezieht sich das Wort entweder auf die serverseitige Zusammenführung der Wartung oder auf den Vorgang des "Terminal-Ready"-Machens der Applikation.

Auf lokaler Seite werden bei Terminals z.Z. Laufwerke und Drucker gemappt.

Die Integration von Software von unterschiedlichen ASPs funktioniert derzeit noch nicht.

Zugriffs- und Lizenzverwaltung

Mit welchen Aufgabenstellungen ist man in der ASP Praxis konfrontiert ?

Bei den Terminals wird die Lizenz meist pro Benutzer erteilt. Das bedeutet, dass ein Benutzer, durch Username und Passwort eindeutig identifiziert, zur gleichen Zeit nur eine Session öffnen kann.

Verrechnungstechnologien

Welche Technologien gibt es in der Praxis und wie funktionieren sie ?

Verrechnungssoftware muss extra dazu erworben werden.
Die Bezahlung der ASP Dienste durch den Kunden erfolgt in der Regel monatlich. Bei Spezialapplikationen muss der Kunde aufzahlen.

Server Herausforderungen

Applikationsprogrammlogik

Welche Aufgaben muss die Applikationsprogrammlogik serverseitig erfüllen ? (bei Websprachen die Applikation, bei Terminals der Terminalserver)

Bei beiden Systemen müssen Vorgänge protokolliert werden und Daten sicher aufbewahrt werden.

Bei den webbasierten Systemen gibt es vor allem die Aufgabe der Sessionverwaltung: Wenn eine Verbindung abbricht, wie und wie lange halte ich die Session-Persistenz (Programm, Daten) aufrecht ?

Bei den Terminals kommen andere Aufgaben hinzu: Migration der Kundendaten, wenn Kunden auf ASP umsteigen, Virenschutz und Verwalten mehrerer User.
Für einzelne Aufgaben kann der Aufwand enorm sein: Wenn ein Benutzer z.B. behauptet, dass eine Winword-Datei fehlt, ist die Suche für die Ursache für das Fehlen, durch das immense Datenaufkommen in der Protokollierung, sehr schwer zu finden.

Verfügbarkeit

Mit welchen Maßnahmen kann Verfügbarkeit erreicht werden ? Ist zwischen ununterbrochenem Applikationszugang (d.h. Stabilität - der Client kann im Fall eines Verbindungsabbruchs dort weitermachen wo er war) und ununterbrochener ZugangsMÖGLICHKEIT (d.h. der Client kann sich zu jeder Tageszeit einloggen) zu unterscheiden ?

Hochverfügbarkeit bedeutet Stabilität: Der User bemerkt einen Ausfall nicht.

Dazu ist es notwendig das Netzwerk ausfallsicher zu gestalten, z.B. Router doppelt anzulegen. Für diejenigen Applikationen, die Daten mit dem Internet austauschen, ist es außerdem wichtig, auch doppelte Internetanbindungen zu haben.

Den Zugang ununterbrochen sicher zu stellen ist hingegen einfach.

Performance und Skalierbarkeit

Anforderungen an Performance und Skalierbarkeit ? (Server, Netz etc.) ?

Bei Websprachen ist die Response-Time relativ unkritisch, da sie zumeist mit der Ausführung von Applikationsfunktionen verknüpft ist. Ich fülle ein Formular aus und zum Abschluss drücke ich den "Submit"-Button. Wie lange es dann dauert ist relativ egal.

Bei den Terminals hingegen ist die Anforderung an die Responsetime sehr hoch. Einige ASPs und AIPs bieten Netzgarantien: Das bedeutet, dass sie für ihr Netz, bei vorgegebenen Einwahlknoten, eine Mindestresponsetime garantieren. Beim Zugriff über das Internet hingegen können solche Garantien nicht abgegeben werden.

Gibt es besondere Aufgaben mit Hinblick auf ASP im Unterschied zu einem Webserver, der nur Content anbietet ?

Ja, in Hinblick auf die Leitungsresponse. Bei den webbasierten Systemen gibt es in der Praxis riesige Belastungspausen. Eine Terminallösung hingegen belastet das Netz fast kontinuierlich.

Service Level Monitoring und Measurement

Was sind die Aufgaben ?

Besonders wichtig sind die OLAs (Operation Level Agreements), primär bei Terminaldiensten. Sie enthalten Regelungen für den Betrieb der Applikationen und der Datenpflege.

Für die Überprüfung und Einhaltung der Operation- und Service Levels gibt es verschiedene Werkzeuge: Wenn die Last steigt, wird z.B. automatisch ein SMS gesendet, ein Telefonanruf ausgelöst oder eine Dialogbox geöffnet. Es muss ein Escalationmanagement geben, und zwar customized ! (d.h. für jeden Kunden individuell). Bei neu eingeführter Software muss es Testumgebungen geben.

Standardüberprüfungen für die SLAs und OLAs müssen automatisiert werden.

Welche Produkte gibt es ?

Es gibt IT-Managementsoftware. Sie ist, gemessen an der Betriebssystemsoftware, sehr teuer. Das prominenteste Beispiel ist "Tivoli" von IBM. Einige wenige Tests kann man auch mit betriebssysteminternen Funktionen durchführen.

Insgesamt belaufen sich die Ausgaben für das IT-Management bei etwa 3 Mio. ATS für 100 Rechner.

Wie funktioniert die Anbindung an die jeweilige ASP Basistechnologie ?

Diese geschieht im Rahmen des Customizing mit großem Aufwand. 1/5 aller Manpower und die Hälfte aller Ausgaben investieren wir in diesem Bereich.

Ein ASP-System muss vor der Freigabe dermaßen konfiguriert sein, dass auch für den Kunden nur das möglich ist, was ausdrücklich erlaubt ist.

Die Anbindung an die Basistechnologie ist noch nicht optimal: Citrix meldet Grundfunktionalität ins Tivoli, diese Informationen sind aber zu wenig.

Bei Windows 2000 gibt es z.B. derzeit noch das Problem, dass es nicht möglich ist, Prozesse auf eine maximale CPU-Nutzung zu begrenzen.

Security

Mir ist bewusst, dass das ein sehr großes Themengebiet ist. Was sind aber die neuen Herausforderungen in Bezug auf ASP ?

Dieser Bereich ist sehr kritisch. Hier investieren wir 1/5 aller Ressourcen. Wichtig, um Hackerangriffe abzuwehren, ist den Inhalt der Verbindungen ständig zu überprüfen. Dazu benötigt man Spezialhardware. Es muss zu jedem Zeitpunkt klar sein, von woher die Anbindung kommt. Deshalb wird oft kein Zugang über das Internet erlaubt, sondern nur über die eigenen Leitungen des ASPs oder AIPs.

Ebenfalls ist die physikalische Sicherheit in den Serverräumen von Relevanz: Es gibt unterbrechungsfreie Stromversorgung (USV). Der Zugang zu den Servern muss abgesichert werden und es gibt Brand- und Bombenschutz. Die Brandlöschung ist problematisch, da zum Löschen der elektronischen Komponenten kein Wasser verwendet werden kann, und daher Löschschaum verwendet werden muss. Menschen müssen innerhalb von drei Minuten den Brandherd verlassen können, da sie sonst an den Löschgasen ersticken könnten.

Aus diesen und vielen anderen Gründen muss es eine Remote-Wartung der Server geben, und zwar von vielen verschiedenen Stellen aus.

Auf Softwareseite müssen die Anwender vor gegenseitiger Datenspionage bewahrt werden. Zum Beispiel müssen Word-Macros unterbunden werden. Der Zugriff auf die Konsole (Eingabeaufforderung unter Windows) und auf Systemverzeichnisse muss gesperrt werden.

Für Fehler in Betriebsystemen, die sich Hacker zunutze machen, ist die Haftung des ASPs oder AIPs jedoch ausgeschlossen.

Client Herausforderungen

Benutzerinterface Programmlogik

Welche Modelle gibt es zur Verarbeitung der Serverdaten (bei Websprachen und Terminals) ?

Terminals stellen die Daten auf dem Bildschirm dar; Citrix in einem eigenen Fenster, WTS zeigt den ganzen Desktop. Alle Daten werden am Server verarbeitet, auch beim Drucken.

Bei den Websprachen kann ich teilweise Logik zum Client verschieben, aber es gibt keine Standards.

Benutzerfreundlichkeit und Anpassbarkeit

Wie benutzerfreundlich kann ein terminalbasiertes System und ein websprachenbasiertes System sein ?

Das Terminalsystem ist beinahe so anpassbar wie ein lokales Betriebsystem. Der "Active Desktop" ist aber beispielsweise untersagt. Das meiste Zubehör (z.B. Calculator) wird standardmäßig aus Sicherheitsgründen nicht zur Verfügung gestellt.

Bei den Websprachen ist fast kein Einfluss auf Anpassbarkeit durch den Anwender gegeben. Nur Opera erlaubt Anpassungen z.B. für Behinderte. *(Anmerkung: Bei der Anpassbarkeit von Websprachen kommt es auf die Fähigkeiten der jeweiligen Applikation an.)*

Wie flexibel (an die persönlichen Bedürfnisse anpassbar) sind die Clienttechnologien ?

Die Anpassbarkeit der Terminalsysteme ist weitaus größer als bei den Websprachen.

Der Begriff "Customizing" bezieht sich i.A. auf einen Mandanten: Dies muss nicht notwendigerweise eine einzelne Person sein, sondern bezieht sich auf einen Personenkreis.

"Mächtigkeit des Clients"

Wie "mächtig" sind die derzeitigen Clienttechnologien ? Welchen Einfluss hat ein Client auf den serverseitigen Ablauf ?

Bei den webbasierten Applikationen ist man auf die Fähigkeiten des Browsers eingeschränkt. Java Applets oder ActiveX-Controls heben diese Schranken zum Teil auf, aber auf Kosten der Security und Performance. Einem ActiveX-Control muss einmal zugestimmt werden, dann hat sie uneingeschränkten Zugriff auf den Client-Rechner. Dasselbe trifft auf SUN JDK zu, jedoch wird hier bei jedem Zugriff nachgefragt.

Bei den terminalbasierten Systemen ist man durch die Fähigkeiten des Terminalclients eingeschränkt. Das Protokoll von Citrix bietet aber sog. "Virtual Channels", über die beliebige Daten übertragen werden können. Ein entsprechendes Client-Modul zur Interpretation dieser Daten kann jederzeit implementiert werden. Allerdings muss jeder virtuelle Kanal registriert und von Citrix freigegeben werden.

Eine Einschränkung von Citrix ist, dass die Verbindung pro Session immer nur zu einem Server steht.

Videos können mit Citrix nur in einem kleinen Bildschirmbereich übertragen werden, nicht große.

Client Sicherheit

Wie sicher ist das Log-In (Passwörter für andere Benutzer am selben Gerät unlesbar etc.) ?

Bei den Websprachen bleibt bei der Verwendung von HTTPS nichts am Client zurück. Über HTTP hingegen sollte kein Log-On abgewickelt werden ! Bei den Terminals kann der Log-On entweder webbasiert über HTTPS erfolgen, oder über terminalbasiertes Windows-Log-On.

Sicherheitslücken in Java-Sandbox oder Citrix ?

Bei den Websprachen haben ActiveX-Controls die komplette Kontrolle über den Rechner, bei Java ebenfalls, wenn es der Benutzer erlaubt.

Bei Citrix oder WTS sind keine Sicherheitslücken bekannt. Bei der Verwendung der Virtual Channels muss einmal zugestimmt werden, dann wird verschlüsselt übertragen.

Wie wird der Fall eines Browserabsturzes behandelt ?

Bei den Websprachen hängt es davon ab, wie viel bereits abgesichert wurde und wie die Session verwaltet wird. Eine generelle Antwort kann hier nicht gegeben werden.

Bei den Terminals ist es möglich, sich innerhalb des Server-Timeouts wieder mit der Session zu verbinden. Dies ist möglich, da auf dem Client keine Sessioninformation gespeichert wird.

ASP - enabling Technologien in der Praxis

Traditionelle Konzepte und Internetsprachen

Grenzen der CGI, Java, JavaScript, PHP, etc.-Technologien ?

Die Grenzen scheinen bereits erreicht.

Windows Terminal Services

Verbreitung bei ASP Diensten ?

Gefühlsmäßig würde ich sagen, dass es in Österreich weiter verbreitet ist als Citrix. Die Unterschiede zu Citrix sind: Mit WTS erzeuge ich einen vollständigen virtuellen Computer, das Clientlaufwerk wird standardmäßig nicht gemappt, und Sound ist derzeit nicht möglich.

Citrix

Verbreitung ?

Aus Kostengründen etwas weniger verbreitet als WTS. Bei einem ASP, der Citrix anbietet, muss man als Endkunde etwa 50-100,- ATS (3,63-7,27 EUR) mehr pro Monat an den ASP bezahlen.

Eignung für ASP ?

Gut.

Enterprise Java Beans

Verbreitung ?

EJB ist ziemlich neu, die Verbreitung bei ASP eher gering, die meisten setzen aber aufgrund von Kosten- und Performancegründen weiterhin auf eigenständige CGI-Lösungen.

Eignung für ASP ?

Das kommt auf die Applikation an.

Zukunftsaussichten ?

EJB wird bestehen bleiben, aber es kommt darauf an, wie Microsoft .NET vom Markt angenommen wird.

.NET – Web Services

Verbreitung ?

Sehr gering, da noch nicht am Markt.

Eignung für ASP ?

Die Web Services sind performanter als EJB, ansonsten bieten sie eine ähnliche Funktionalität.

Zukunftsaussichten ?

Die Zukunftsaussichten von .NET im ASP-Bereich sind noch unbekannt.

Proprietäre Lösungen

Welche technischen Wege beim Anbieten von ASP Lösungen werden z.Z. noch eingeschlagen ?

Weit verbreitet sind die Shop-Lösungen, Calendaring, Messaging (wie E-Mail etc.). In allen Fällen, wo Semantik benötigt wird, werden webbasierte Lösungen gerne verwendet.

Abkürzungsverzeichnis

ACL	Access Control List
ADSL	Asynchronous Digital Subscriber Line
AG	Aktiengesellschaft
AH	Authentification Header
AIP	Adaptive Internet Protocol
AIP	Application Infrastructure Provider
API	Application Program Interface
ASP	Active Server Pages
ASP	Application Service Provider
ASP	Application Service Providing
ATM	Asynchronous Transfer Mode
ATS	Austrian Schillings
B2B	Business-to-Business
BDSG	Bundesdatenschutzgesetz
BGB	Bürgerliches Gesetzbuch
BIOS	Basic Input / Output System
bspw.	beispielsweise
bzw.	beziehungsweise
CA	Certification Authority
CAD	Computer Aided Design
CAST	Carlisle Adams and Stafford Tavares
CD	Compact Disc
CGI	Common Gateway Interface
CLR	Common Language Runtime
COM	Communication
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CRM	Customer Relationship Management

CSS	Cascading Style Sheets
d.h.	das heisst
DCOM	Distributed Component Object Model
DEM	Deutsche Mark
DES	Data Encryption Standard
D-H	Diffie-Hellman
DMZ	Demilitarized Zone
DNS	Domain Name System
DoS	Denial of Service
DOS	Disk Operating System
DSA	Digital Signature Algorithm
DSL	Digital Subscriber Line
DTD	Document Type Definition
DTP	Desktop Publishing
ECDH	Elliptic Curve Diffie Hellman
ECMA	European Computer Manufactures Association
EEPROM	Electrically Erasable Programmable Read Only Memory
EJB	Enterprise Java Beans
EPROM	Erasable Programable Read Only Memory
ERM	Enterprise Relationship Management
ERP	Enterprise Relationship Planning
ESP	Encapsulating Security Payload
et al.	et altera
etc.	et cetera
EUR	Euro
ev.	eventuell
EZB	Europäische Zentralbank
FTP	File Transfer Protocol
GB	Gigabyte
gem.	gemäß
GmbH	Gesellschaft mit beschränkter Haftung
GUI	Graphical User Interface
GXA	Global XML Architecture
HSP	Hosting Service Provider
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol

HTTPS	Hypertext Transfer Protocol Secure
i.A.	im Allgemeinen
i.d.F.	in der Fassung
i.s.d.	im Sinne des/der
i.S.v.	im Sinne von
ICA	Independent Computing Architecture
ID	Identität
IDEA	International Data Encryption Algorithm
IDS	Intrusion Detection System
IE	Internet Explorer
IETF	Internet Engineering Task Force
IIOP	Internet Inter-Object Request Broker Protocol
IKE	Internet Key Exchange
IL	Intermediate Language
inkl.	inklusive
IP	Internet Protocol
IPSec	Internet Protocol Security
IPX	Internetwork Packet Exchange
ISDN	Integrated Services Digital Network
ISO	International Standardization Organisation
ISP	Internet Service Provider
ISV	Independent Software Vendor
IT	Information Technology
ITIL	Information Technology Infrastructure Library
ITU	International Telecommunications Union
J2EE	Java 2, Enterprise Edition
JDBC	Java Database Connectivity
JIT	Just-In-Time
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
Kb	Kilobyte
Kb/s	Kilo bytes per second
Kbps	Kilo bits per second
KMU	Kleine und mittlere Unternehmen
LAMP	Linux-Apache-MySQL-PHP

LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LPT	Line Print Terminal
LTPA	Lightweight-Third-Party Authentication
MAN	Metropolean Area Network
MB	Megabyte
MCCS	MiiCurrencyConverterService
Mill.	Millionen
MIME	Multipurpose Internet Mail Extensions
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
NAT	Network Address Translation
NetBEUI	NetBIOS extended user interface
NetBIOS	Network Basic Input/Output System
NFS	Network File System
Nr.	Nummer
NSP	Network Service Provider
ODBC	Open DataBase Connectivity
OLA	Operation Level Agreement
ÖNB	Österreichische Nationalbank
OS	Operation System
OSI	Open Systems Interconnection
PC	Personal Computer
PDA	Personal Digital Assistant
PDF	portable document format
PHP	Personal Homepage Tools (Hypertext Preprocessor)
PKI	Public Key Infrastructure
QoS	Qualities of Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RC4	Rivest's Cipher 4
RC5	Rivest's Cipher 5
RDBMS	Relational Database Management System
RDP	Remote Desktop Protocol
RISC	Reduced Instruction Set Computer
RMI	Remote Method Invocation

ROI	Return Of Investment
ROM	Read Only Memory
RPC	Remote Procedure Call(ing)
RSA	Rivest – Shamir – Adleman
SA	Security Association
SGML	Standardized Generalized Markup Language
SLA	Service Level Agreement
SME	Small and medium Enterprise
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
sog.	sogenannte
SPD	Security Policy Database
SPX	Sequenced Packet Exchange
SQL	Structured Query Language
SSH	Secure Shell
SSI	Server Side Includes
SSL	Secure Sockets Layer
TCO	Total Cost of Ownership
TCP	Transmission Control Protocol
TDDSG	Teledienstedatenschutzgesetz
tlw.	teilweise
u.ä.	und ähnliche
u.U.	unter Umständen
u.v.a.m.	und viele(s) andere mehr
UDDI	Universal Discovery Description Language
UDP	User Datagram Protocol
UID	User Interface Desgin
UrhG	Urheberrechtsgesetz
URL	Unified Resource Location
US	United States
USB	Universal Serial Bus
USD	United States Dollars
USV	Unterbrechungsfreie Stromversorgung
VGA	Video Graphics Array
VM	Virtual Maschine

VPN	Virtual Private Network
WAN	Wide Area Network
WAP	Wireless Application Protocol
WML	Wireless Markup Language
WS	Web Services
WSDL	Web Service Description Language
WTS	Windows Terminal Services
WWW	World Wide Web
XML	eXtensible Markup Language
XSL	eXtesible Stylesheet Language
z.B.	zum Beispiel
z.T.	zum Teil
z.Z.	zur Zeit

Literaturverzeichnis

[app2web 2002] Application2Web
Stand der Technik
Projektzusammenarbeit zwischen dem Forschungszentrum Informatik (FZI) Holger Bär und Frauenhofer-Institut für experimentelles Software Engineering (IESE) Jean-François Girard sowie zahlreichen Unternehmen
http://app2web.fzi.de/themen/ap1.xml

[Arthur D Little 2000] Arthur D Little
Risks and opportunities of Application Service providing
November 2000
eBusiness Newsletter

[ASP I.C. 2000] ASP Industry Consortium
An Overview of "Dispute Avoidance and Resolution Best Practices in the ASP Industry" White Paper
Best Practices Committee, Application Service Provider Industry Consortium, In conjunction with the WIPO Arbitration and Mediation Center
http://www.asp.com/pdf/DARToverview.pdf

[ASP I.C. 2001] ASP Industry Consortium
all about asp-Resources-FAQs
http://www.allaboutasp.com/faqs.cfm

[Bauer 2001] Stefan Bauer, Josef Weinzierl
Application Service Providing als alternative Form der Software Distribution
Arbeitsbericht der Veranstaltung "Teledienste - Trendanalyse und Bewertung" am Lehrstuhl für Allgemeine und Industrielle Betriebswirtschaftslehre der Technischen Universität München, März 2001

[BEA 2001] BEA Systems Inc.
BEA WebLogic Server and BEA WebLogic Express - Introduction to BEA WebLogic Server 6.1
http://www.bea.com

[Berheide 2000] Thomas Berheide
Eine Benchmarkstudie über die Application-Service Providing Branche
Forschungsbericht, im Rahmen des Seminars zur Distribution und Handel: Internationale Erfolgsforschung im Wintersemester 2000/01, Universität Münster
http://www.wiwi.uni-muenster.de/~02/download/seminar/7.pdf

[Bettinger 2001] Torsten Bettinger, Michael Scheffelt
Application Service Providing: Vertragsgestaltung und Konflikt-Management
Computerrecht 11/2001
http://www.bettinger.de/urteile/cr_0111_bettinger.pdf

[BinTec 2001] BinTec
IPSec
August 2001
www.bintec.de/de/prod/pdf/ds/DS_sw_IPSec_de.pdf

[Bird 2001] Tina Bird
Vrtual Private Networks - Frequently Asked Questions
Letzte Aktualisierung: 19. August 2001
http://kubarb.phsx.ukans.edu/~tbird/vpn/FAQ.html

[Borland 2002] Borland Software Corporation
Borland Unveils Product Strategy for Microsoft® .NET? Platform
Ankündigung vom 12. Februar 2002
http://www.borland.com/net/

[Brandstätter 2001] Nicole Brandstätter, Christian Haim, Mertin Kottrasch
SOAP-Simple Object Access Protocol
Seminararbeit WS 2001 / 2002, Universität Linz, Institut für Wirtschaftsinformatik, Abteilung Software Engineering
http://www.swe.uni-linz.ac.at/teaching/lva/ws01-02/projektstudium/PROST246.535/SOAP-Seminararbeit.doc

[Cherry Tree 1999] Cherry Tree & Co.
"Application Service Providers" (ASP)
Spotlight Report, www.cherrytreeco.com

[Chiu 2001] Edmond Chiu
Guidelines for Security at the Application Service Provider and Hosting Service Organization
19. Juli 2001
http://rr.sans.org/securitybasics/app_service.php

[Ciresa 2002] Meinhard Ciresa
E-Mail-Schriftverkehr am 21. Jänner 2002 über *rechtliche Aspekte von ASP in Österreich.*
ra@ciresa.at
http://www.ciresanet.at

[Citrix 2002] Citrix
Citrix Homepage
http://www.citrix.com

[Compaq 2001] Helmut Plickert
ASP: Software Mieten - Softwarenutzung via Internet vor dem Durchbruch
Compaq Computer GmbH
Vortragsfolien DECUS Symposium Berlin 24.-26. April 2001

[CompTIA 2001] Computing Technology Industry Association
Specification for Best Practices in Electronic Service Provisioning
April 2001, herausgegeben von Clicksure

[CompTIA 2002] Computing Technology Industry Association
White Paper on xSPertise
http://www.comptia.org/services/trustmark/xspertise/whitepaper.htm

[Corecom 2001] Core Competence, Inc.
VPNs: Virtually Anything?
Copyright 2001
http://www.corecom.com/html/vpn.html

[c-quential 2000] c-quential-An Arthur D. Little Company
Trends in the ASP market-national and international development
Präsentation zur ASP Konferenz vom 30. November 2000 in Wien, zu finden auf der CD zur Konferenz

[DeGiglio 2000] Maria DeGiglio
IBM ASP Prime for Developers: Positioning Your Business for Success in the ASP Marketplace
November 2000,
http://www.developer.ibm.com/aspprime/library.html

[Diebold 1999] Diebold Management Institut
ASP-Application Service Providing, Seminar (Inhalt, Ziele und Programm)
http://www.diebold.de/veranstalt/pdf/ASP.pdf

[Dunigan 2002] Tom Dunigan
Network Performance links
Sammlung von Links zum Thema Netzwerkperformance.
Letzte Aktualisierung: 22.Februar 2002
http://www.csm.ornl.gov/~dunigan/netperf/netlinks.html

[ECO 2000] The EMC, Cisco, Oracle E-business "ECOstructure" initiative
E-Business Foundations Without Limits
A Business White Paper

[ECO 2001] The EMC, Cisco, Oracle E-business "ECOstructure" initiative
Accelerated Blueprint
Version 2.0
http://www.eecostructure.com/blueprint.html

[Farleit 2000] Farleit Limited
Anatomy of an ASP-Defining the new genus
White Paper, Mai 2000
http://www.aspnews.com/pubs/anat0005.pdf

[FHZ 2000] Adrian Fellmann, Tobias Frommenwiler, Christoph Rechberger, Andreas Woytschak
Outsourcing von Informatik-Lösungen
Projektarbeit unter der Leitung von Dozent Dr. Carlo Ronca, 5. Mai 2000
http://www.frommenwiler.ch/hsw/pa/projektarbeit.pdf

[Fischer 2001] Martin Fischer
Serverseitige Programmierung von Webservern
Universität Basel, Seminar zur Wirtschaftsinformatik, Prof. Dr. M. Lusti, betreut von Patrick Wirz
http://www.wwz.unibas.ch/wi/lehre/swi/Seminararbeiten/mfi.pdf

[Franssen 2000] Stephan Franssen
Application Service Provision (ASP) as a solution for Financial Institutions: the vertical market perspective
Seminararbeit im Rahmen des Seminars Wirtschaftsinformatik, European Business School, Schloß Reichhartshausen am Rhein

[Gillan et al. 1999] Clare Gillan, Steve Graham, Mark Levitt, John McArthur, Steve Murray, Vernon Turner, Rick Villars und Meredith McCarthy Whalen
The ASPs' Impact on the IT Industry: An IDC-Wide Opinion
Bulletin, IDC International Data Corporation,
http://www.tds-global.com/images/asp_idc_studie.pdf

[grouptech 2002] Simon Hirth, Damian Seibert
ASP Softwarehosting-Vergleichsstudie
GROUP Technologies AG
http://www.asp4you.de/

[Gruber 2000] Karin Gruber
ASP-SAP Lösungen mieten statt kaufen
SAP Österreich GmbH, Präsentation zur ASP Konferenz vom 30. November 2000 in Wien, zu finden auf der CD zur Konferenz

[Gschwend 2001] Matthias Gschwend
Potential des Electronic Business in der Inkassobranche
Lizentiatsarbeit, Universität Freiburg (Schweiz)
30. Juni 2001
http://www2-iiuf.unifr.ch/is/PDF/Gschwend2001DA.pdf

[Herzwurm 2001] Georg Herzwurm, Harald Schlang
Technologien zur Programmierung von Internet-Anwendungen
Präsentationsfolien zur Vorlesung "Programmierparadigmen" im WS 01/02
Universität zu Köln, Lehrstuhl für Wirtschaftsinformatik, Systementwicklung, Prof. Dr. Werner Mellis
http://www.systementwicklung.uni-koeln.de/lehre/awinfo1/dok-ws00-01/1-4-2-technologien-zur-program-von-internet-anwendungen-2f.pdf

[Hollander 2002] Yona Hollander
Systems Shield -- The Future of Web Security White Paper
Entercept Security Technologies
http://www.counterstrike.com/ss-white.html

[Hußmann 2001] Heinrich Hußmann
Web-Komponenten mit Java
Präsentationsfolien zur Vorlesung "Softwarekomponenten"
Technische Universität Dresden, Lehrstuhl Softwaretechnologie im Institut für Software- und Multimediatechnik (SMT)
http://www-st.inf.tu-dresden.de/sk/vorlesung/sk4_extra.pdf

[IBM 2001] IBM Corporation
WebSphere Application Server, Version 4.0-Integrating data and transactions for agile e-business
htttp://ibm.com/software/webservers/appserv

[IC 2001] InterConnection Consulting Group Wien
ASP in Österreich: Convenience entscheidet über einen Milliardenmarkt
Presseaussendung 2001, zu finden auf www.aspgroup.at

[Igler 1999] Martin Igler
Mieten statt kaufen-Abrufbare IT-Leistung
Lünendonk im Auftrag der TDS,
http://www.luenendonk.de/archiv/ASP_fibel.PDF

[Infonomics 2001] Thomas Nebe
Service Management Whitepaper
April 2001, Infonomics-Conulting

[iStart 2002] iStart
Internet Security - Protecting your company's most valuable asset-information
New Zealand's e-Business Portal
http://www.istart.co.nz/computer_security.htm

[JDJ 2001] Java Developer's Journal
Umfrage zum besten Java Applikationsserver
Stand: 17. Jänner 2001
http://www2.sys-con.com/java/readerschoice2001/liveupdateappserver2.cfm

[Kehrer 2000] Bernd Kehrer, Uwe Döttge, Helmut Jansen, Uwe von Lukas
Innovative Strategien für das Software-Marketing - Software on Demand / Application Service Providing (ASP)
2000, TU Berlin
http://www.rostock.igd.fhg.de/ZGDV/files/ZGDV/Abteilungen/zr1/Veroeffentlichunge/InnovationsForum2000/document.pdf

[Kennedy 2000] Traver Gruen-Kennedy, Lisa Paul
Executive Interview with Traver Gruen-Kennedy, Topic: The ASP Industry Consortium
September 2000, http://www.aspisland.com/interviews/

[Kolk 2001] Daniel Kolk
Rechtliche Aspekte des Application Service Providing-vertragstypologische Einordnung und Datenschutz
Hausarbeit zur wirtschaftswissenschaftlichen Zusatzausbildung, Rechts- und wirtschaftswissenschaftliche Fakultät, Universität Bayreuth.
http://www.stud.uni-bayreuth.de/~a4856/jura/arbeiten/asp.html

[Kopetz 1996] Hermann Kopetz
Echtzeitsysteme
Technische Universität Wien
Mai 1996
Skriptum zur Vorlesung "Echtzeitsysteme"

[Luckhardt 2002] Norbert Luckhardt
Virenschutz ist mehr als Software
http://www.security-forum.de

[Microsoft 2002a] Microsoft Corporation
Microsoft Mobile Information Server
http://www.microsoft.com/miserver
http://msdn.microsoft.com/vstudio/technical/articles/mobilewebforms.asp

[Microsoft 2002b] Microsoft Corporation
Implementing Sun Microsystem's Java Pet Store J2EE Blueprint Application using Micorosft .NET
Version 1.5, November 2001
http://www.gotdotnet.com/team/compare/Petshopdoc.zip

[Microsoft 2002c] Microsoft Corporation
Terminal Services
http://www.microsoft.com/windows2000/technologies/terminal/default.asp

[Microsoft 2002d] Microsoft Corporation
My Services
http://www.microsoft.com/myservices/default.asp

[Microsoft 2002e] Microsoft Corporation
WS-Security Specification Index Page
Jänner 2002
http://msdn.microsoft.com/library/en-us/dnglobspec/html/wssecurspecindex.asp

[ModusNovo 2002a] ModusNovo
Service Usage and Billing for the Internet: Technical Issues
http://www.modusnovo.com/wp/BillingTech310100.pdf

[ModusNovo 2002b] ModusNovo
Service Usage and Billing for the Internet: Business Issues
http://www.modusnovo.com/wp/BillingBusiness310100.pdf

[Momut 2000] Raymont Momut, Peter Geier, Lee Crawley
DSL Security Whitepaper
Mai 2000
http://www.adsl.com/aboutdsl/security_whitepaper.doc

[Mono 2002] Ximian, Inc.
"*Mono::" - Projekt*
http://www.go-mono.com

[MSN 2002] Microsoft Corporation
MSN Explorer
http://explorer.msn.at

[Münz 2001] Stefan Münz
SELFHTML
Version 8.0 vom 27. Oktober 2001
http://selfhtml.teamone.de

[Neefe 2000] Achin von Neefe
Security Issues for Application Service Providers
November 2000
http://rr.sans.org/start/app_service.php

[Nieh 2000] Jason Nie, S. Jae Yang, Naomi Novik
A Comparison of Thin-Client Computing Architectures
Technical Report CUCS-022-00, November 2000
Network Computing Laboratory, Columbia University
http://www.ncl.cs.columbia.edu/publications/cucs-022-00.pdf

[Paiha 2001] Dominik Paiha
Interview am 8. Dezember 2001
Thema: *Technische Aspekte von ASP*

[Plessl 2001] Christian Plessl, Erik Wilde
Dienstbare Geister
März 2001
http://www.heise.de/ix/artikel/2001/03/088/

[Rupp 2002] Christoph Rupp
Application Server vs. Application Service Providing-Von der Komponentenwelt zur Service Infrastruktur
Seminararbeit, Johann Wolfgang Goethe-Universität, Frankfurt am Main
14. Jänner 2002
http://interactive.wiwi.uni-frankfurt.de/iwi/veranstaltung/SBWL-WS0102/downloads/5_rupp.pdf

[Saly 1999] Hendrik Saly
Datenbankanbindung von Web-Servern in Java
Fachhochschule Offenburg, Hochschule für Technik und Wirtschaft, Studienarbeit aus Medien und Informationswesen
16. August 1999
http://info.mi.fh-offenburg.de/html/studenten/galerie/studienarbeiten/dbjava_more.html

[SAP 1999] SAP
SAP annual report 1999
http://www.sap.com/germany/investor/pdf/f20f99.pdf

[Sattler 2001] Kai-Uwe Sattler
Kopplungstechniken und Zugriffsschnittstellen
Präsentationsfolien zur Vorlesung "Internet-Datenbanken"
Universität Magdeburg
http://wwwiti.cs.uni-magdeburg.de/~sattler/lectures/webdb07.pdf

[Schaub 2000] Martin Schaub
Microsoft .NET - Einblick und Überblick
Studienarbeit, Berufsakademie Lörrach,
http://www.machtien.de/Studienarbeit/Microsoft%20DotNET%20Final.pdf

[Schnabel 2002] Patrick Schnabel
VPN - Virtual Private Network
eOnline.de
http://www.e-online.de/sites/kom/0512041.htm

[SCN Education 2000] SCN Education B.V. (Herausgeber)
ASP-Application Service Providing
1. Ausgabe 2000, ISBN 3-528-03148-4

[Stahlknecht 2000] Peter Stahlknecht
Aktueller Stand und Entwicklungstendenzen im IT-Outsourcing
Fachtagung der GOR-Arbeitsgruppe Wirtschaftsinformatik, Institut für Informationsmanagement und Unternehmensführung, Universität Osnabrück

[Stary 1996] Christian Stary
Interaktive Systeme
Software-Entwicklung und Software-Ergonomie
2. Auflage, Vieweg Verlag

[Steffen 1999] Günter Steffen
Blick zurück nach vorn-Vom Outsourcing zum Application Service Provider
TDS
http://www.tds.de/images/buch_koehler_frost.pdf

[Summit 1999a] Summit Strategies
Internet Applications Hosting: The Top Ten Business Considerations for ISVs
October 1999
http://www.developer.ibm.com/aspprime/library.html

[Summit 1999b] Summit Strategies
Internet Applications Hosting: The Top Ten Technical Considerations for ISVs
October 1999
http://www.developer.ibm.com/aspprime/library.html

[Tao 2000] Lixin Tao
Application Service Provider Model: Perspectives and Challenges
http://www.cs.concordia.ca/~faculty/lixin/download/asp.pdf

[Thai 2002] Thuan L. Thai, Hoang Larn
.NET Framework Essentails
2nd Edition Februar 2002

[Tramm 2000] Gerrit Tramm
Business concepts and strategies for the ASP market
Institute of Information Systems, Humboldt University,
http://miro.wiwi.hu-berlin.de/~gtamm/gkvi/proposal/wulkow-April-2000.PDF

[W3C 2000a] W3C
Simple Object Access Protocol (SOAP) 1.1
W3C Note, 8. Mai 2000
http://www.w3.org/TR/SOAP/

[W3C 2000b] W3C
Extensible Markup Language (XML) 1.0 (Second Edition)
W3C Recommendation, 6. Oktober 2000
http://www.w3.org/TR/REC-xml.html

[W3C 2001] W3C
Web Service Description Language (WSDL) 1.1
W3C Note, 15. März 2001
http://www.w3.org/TR/wsdl

[Wagner 2001] Dietrich Wagner
User experiences with ASP: myths and realities
18.6.2001, Universität Bayreuth Rechts- und Wirtschaftswissenschaftliche Fakultät, Seminararbeit im Rahmen des Seminars Information Systems Outsourcing in the Age of the New Economy: Contemporary Issues and Future Directions
http://wi.oec.uni-bayreuth.de/veranstaltung/01_ss/seminar/paper/09_wagner.pdf

[Wurzenberger 2000] Georg Wurzenberger
Aspekte moderner Web-Applikationen - Gegenwärtige Technologien und Server im Uberblick, sowie in der Praxis
Diplomarbeit aus Telematik
Technische Universität Graz, März 2000
http://www.iicm.edu/thesis/gwurzenberger.pdf